World Applied Sciences Journal 29 (9): 1156-1161, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.29.09.1428

Parallel Architecture for the VLSI Implementation

A. Geetha

Department of ECE, Bharath Univeristy, Chennai - 73, India

Abstract: Error detection and correction plays a very important role in data communication. Various codes such as convolutional and block codes are available for the purpose of error detection and correction. Among the block codes Reed-Solomon code provides several advantages. Reed-Solomon codes are powerful error-correcting codes that finds wide applications in many fields. The soft-decision decoding of Reed-Solomon codes provides reliable information from the channel into the decoding process. Many soft-decision decoding algorithms are available. Among them the Koetter-Vardy algorithm is used in this paper. The gain of soft-decision decoding algorithm is several db greater than hard-decision decoders. The algorithm consists of three parts namely a soft decision front end, an interpolation processor, and a factorization step. Among the three the interpolation step is little complicated. This paper provides a parallel architecture for the VLSI implementation of the interpolation processor.

Key words: Reed-Solomon codes · Soft-decision decoding · Interpolation

INTRODUCTION

To meet the demands of today's continually evolving communication systems, digital communication finds a wide application in many fields such as base-band processing, digital IF processing etc. An important function of any modern digital communications system is error control coding. Such coding in the field of communication deals with techniques for detecting and correcting errors in a signal. Though used in a variety of systems, error control coding is especially useful in wireless communication system. Such systems typically operate with a low signal-to-noise ratio (SNR) and suffer from distortion because of a multipath channel. The harsh wireless environment means that the received signal is prone to errors.

Typical communication systems use several codes that are suited to correcting different types of errors. Reed-Solomon (RS) codes are the most powerful in the family of linear block codes and are arguably the most widely used type of error control codes. The fundamental concept of error control coding is the addition of redundancy to a signal at the transmitter and the exploitation of that redundancy at the receiver to detect and/or correct errors. The inclusion of redundancy in the transmitted signal results in a coding signal consisting of more bits than the original uncoded signal. The trade-off for this overhead is the ability to detect, and possibly correct errors at the receiver. The performance improvement that occurs when using error control coding is often measured in terms of coding gain [1].

Reed-solomon Code: Reed-Solomon are block-based error correcting codes that can be found in many digital communication standards. It finds a wide range of applications in many areas such as storage devices, wireless or mobile communications, satellite communications, digital television, high speed modems.

The Reed-Solomon encoder takes a block of digital data and adds extra "redundant" bits. Errors occur during transmission or storage for a number of reasons such as noise or interference, scratches on a CD. The Reed-Solomon decoder processes each block and attempts to correct errors and recover the data. The number and type of errors that can be corrected depends on the characteristics of the Reed-Solomon code.

Properties of Reed-solomon Codes: The Reed-Solomon code is specified as RS(n,k) with s-bit symbols. This means that the encoder takes k data symbols of

Corresponding Author: A. Geetha, Department of ECE, Bharath Univeristy, Chennai - 73, India.

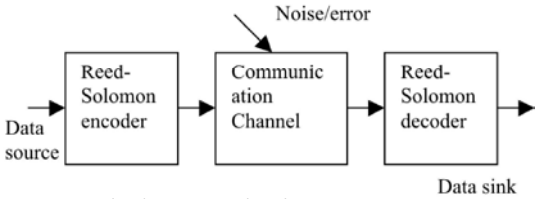

Fig. 1: A typical communication system

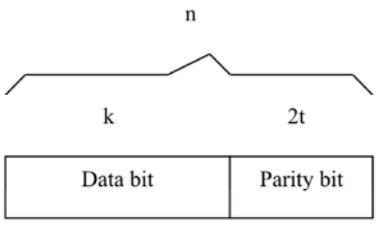

Fig. 2: A typical Reed-Solomon code word

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 [2].

A popular Reed-Solomon code is RS (255, 223) with 8-bit symbols. Each codeword contains 255 code word bytes, of which 223 bytes are data and 32 bytes are parity. For this code:

$$n = 255, k = 223, s = 8$$
$$2t = 32, t = 16$$

The decoder can correct any 16 symbol errors in the code word. In other words errors in up to 16 bytes anywhere in the codeword can be automatically corrected. The amount of processing "power" required to encode and decode Reed-Solomon codes is related to the number of parity symbols per code word.

Soft-Decision Decoding of Reed-solomon Codes: A Reed-Solomon code RS(n,k) encodes k message symbols drawn from a finite field with q elements, GF(q), into a code word of n<=q message symbols also drawn from GF(q). The encoder is an evaluation map of a message polynomial

$$f(x) = f_0+f_1x+f_2x^2+\ldots\ldots\ldots+f_{k-1}x^{k-1} \quad (1)$$

where the coefficients fi are the k message symbols. If the message polynomial f(x) and n distinct values $\{a_1, a_2, a_3, \ldots\ldots\ldots a_n\}$ are given then the Reed-Solomon code word is

$$C = (sf(a_1),f(a_2),\ldots\ldots\ldots,f(a_n)) \quad (2)$$

usually n = q-1 and there are q-1 non zero evaluation elements of GF(q).

Interpolation Based Soft-decision Decoding: The code word is transmitted over a noisy channel and the corrupted word is received. The soft-decision Koetter-Vardy decoding algorithm is a style of decoder called list decoder. The list decoding task is to find a set of code words that disagree with the received word 'y' in at most places [3]. The decoding process is done through some sort of interpolation. Soft-decision decoding involves weighted interpolation, where the weights are derived from the soft reliability information.

Koetter-vardy Algorithm: The soft information is received in the form of a q x n reliability matrix. The steps of the interpolation-based Koetter-Vardy algorithm are as follows

Soft-decision front end: Using the soft reliability information, derive positive integer weights for each points. The KV algorithm computes weights proportional to the soft information [4, 5]. These weights are stored in a multiplicity matrix.

Interpolation: Find the bivariate interpolation polynomial P(x,y) of minimal (1,k-1) that passes through all the points with prescribed multiplicities.

Factorization: Find all the factors of P(x,y) of the form y-f(x) with deg f(x)<k. The output of the algorithm is the list of code words that corresponds to these factors.

If we find more than one possible code word, the soft reliability information about y can be used to select from the list. Of the three main components of the algorithm, the most computationally expensive one is the bivariate interpolation step [6]. Motivated by a VLSI implementation view point, we propose a improvement to the interpolation algorithm that uses a transformation of the received word to reduce the number of iterations.

Architecture for Interpolation: Efficient interpolation is done using an algorithmic transformation known as reencoding [6, 7]. The reencoding reduces the number of interpolation points. The idea of reencoding is to transform the interpolation problem into one that is easier to solve. The code word C is transmitted through a noisy channel. The hard-decision vector r is extracted from the reliability matrix, where r = c+e, where e is an error vector.

The first step in reencoding is to partition the received symbol in r into two sets, U(unreliable) and R(reliable). The set R consists of the k most reliable symbols, where the reliability information can be derived from the multiplicity matrix. Now systematically encode the symbols in R so that they appear in the reencoded code word in the same position that they appeared in r. Now the difference between r and the reencoded code word gives a code word that is corrupted by the same error pattern as in r. However the k symbols of the difference are zero. The zero symbols corresponds to k interpolation points with zero y-component.

Algorithm1: Interpolation algorithm to find a polynomial p(x,y) with minimal $(1,w_y)$ weighted degree and maximum y-degree d_y.

Input:A set of triples $\{(x_j,y_j,m_j)\}$
Init:
G← $\{g_0(x,y)=y^0, g_1(x,y)=y^1 \ldots, g_{dy}(x,y)=y^{dy}\}$
D← {d0=0,d1=wy,d2=2wy,……ddy=dywy}
for each point (xj,yj) with multiplicity mj>0 do
for μ ←0 to mj-1do
for *v* ←0 to mj-1-μ do
O = $\{i: g_i^{[u,v]}(x_j,y_j)\neq 0\}$
d = argmini e O {dj}
for i=0,……..,d_y do
if i ≠ δ then
$g_i(x,y) \leftarrow g_\delta^{[\mu,v]}(x_j,y_j).g_i(x,y)$-gi[μ,v]$(x_j,y_j).g\delta(x,y)$
end if
end for
gδ(x,y) $(x-x_j)$ gδ(x,y)
dδ ← dδ+1
end for
end for
end for
δ = arg min $\{d_i\}$,I = 0,……,dy

Output: p(x,y) = gδ(x,y)

VLSI Architecture for Interpolation: The bivariate interpolation step is the most computationally demanding step in the Koetter-Vardy algorithm.

Computation in the Interpolation Algorithm: The cost C of interpolation is the number of iterations of the algorithm given above. The upper bound on the y-degree of the polynomial produced by the algorithm is

$$d_y = \left\lfloor \frac{1+\sqrt{1+\frac{sc}{k-1}}}{2} \right\rfloor - 1 \quad (3)$$

and the upper bound on the x-degree is

$$d_x = \left\lfloor \frac{C}{d_y+1} + \frac{d_y}{2}(k-1) \right\rfloor \quad (4)$$

where C represents the cost of the interpolation before reencoding. The interpolation algorithm updates a set of polynomials which we refer to as the scratch polynomial. Only one of the scratch polynomial with minimum weighted degree will be chosen as the ultimate answer. There are five computational tasks in a given iteration of the interpolation algorithm.

HASSE Derivative: The Hasse derivative for the (dy+1) scratch polynomials is given as

☻(5)

MIN: The minimum weighted degree for the set of all scratch polynomials with non-zero Hasse derivative is calculated.

ROWOP: The scratch polynomials are updated except for the one with minimum weighted degree.

ZERO: A zero is added to the scratch polynomial with the minimum weighted degree.

DEG: The weighted degree of the scratch polynomial is incremented.

The ROWOP, ZERO, HASSE are generally polynomial operations and they are expensive to implement due to the length of bivariate polynomials. MIN and DEG operates on small set of integers and are straight forward to implement. Parallel operations can even speed up the polynomial operations.

Polynomial Representation: Polynomial representation is suitable for parallel computation. Here every possible term is represented whether or not its coefficients is zero. The polynomial addition is done point wise by adding like terms. A parallel representation becomes feasible when

using the reduced polynomials that results from the reencoding and change of variable techniques.

Monomial ordering is another type of polynomial representation in which the polynomials are stored in a weighted degree lexicographic order. Therefore we are free to choose any ordering that is convenient for our architecture. Monomials of like y-degree are grouped together [1, 6].

The bivariate polynomial $g_i(x,y)$ can be represented as a univariate polynomial in y, where the coefficients of the powers of y are univariate polynomials in x.

$$g_i(x,y) = \sum_{v=0}^{d_y} w_i^v(x) y^v, 0 \le i \le d_y \tag{6}$$

The univariate polynomial can be represented as

$$w_i^v(x) = \sum_{u=0}^{L_v} w_{i,u}^v x^u. \tag{7}$$

Length of each univariate polynomial is

$$L_v = \left(d_x - mk + 1 + v\right) \tag{8}$$

And the total length of bivariate polynomial is

$$L = \sum L_v \tag{9}$$

The set of scratch polynomials G can be written in a matrix representation as

$$W = \{W_0, W_1, \ldots\ldots\ldots W_{dy}\} \tag{10}$$

All polynomial computation in the interpolation algorithm reduces to combinations of a simple set of basic operations on the rows of each matrix.

Addition of Two Polynomials:

$$w_a^v(x) + w_b^v(x) : \vec{w}_a^v \leftarrow \vec{w}_a^v + \vec{w}_b^v. \tag{11}$$

Multiplication of a polynomial by x

$$xw_a^x(x) : \vec{w}_a^v \leftarrow (0, w_{a,0}^v, w_{a,1}^v, w_{a,L-2}^v). \tag{12}$$

Multiplication of a polynomial by a scalar

$$xw_a^?(x) : \vec{w}_a^v \leftarrow (0, w_{a,0}^v, w_{a,1}^v, \ldots\ldots, w_{a,L-2}^v). \tag{13}$$

Evaluation of a polynomial

$$w_a^v(f) = \sum w_{a,u}^v f^u. \tag{14}$$

Parallel Architecture for the Polynomial Updates: The entries of the matrix can be represented by a dependence graph (DG). The dependence graph for the ROWOP operation consists of polynomial addition and scalar multiplication. The polynomial parallel architecture for the polynomial operations is shown below

A linear array of (d_y+1) processing elements(PEs) updates the (d_y+1) scratch polynomials in parallel. The polynomials can be stored in SRAM arrays and the operations are done serially, term by term. The length of the polynomial is much larger than the maximum y degree. Therefore the degree of parallelism is high giving a large speed-up factor. The L PEs are subdivided into (dy+1) independent arrays. The resulting architecture for ROWOP and ZERO are shown below.

Both the operations can be executed in a single clock cycle, assuming that the scalar variables are available [8]. The critical path is the delay of one Galois field multiplier and one Galois field adder.

Differential interpolation is an interpolation schedule [4] that increases the multiplicities of individual points to improve the decoding performance. Some of the Hasse derivatives might be satisfied in previous stage of decoding [9]. To satisfy additional constraints in a subsequent stage another algorithm can be used to bring a point from arbitrary multiplicity S to multiplicity S+1. The algorithm is shown below. Alg

Interconnection Network: The overall architecture for the parallel bivariate polynomial processor is shown below.

To evaluate a Hasse derivative the XPROCs each execute a polynomial evaluation, the results of which are sent to a top-level processor, YPROCs which contain a YHASSE module, circuits for calculating MIN and DEG, and a control unit. Once a HASSE derivative of a bivariate polynomial has been found, it is broadcast to each of the XPROCs simultaneously for use in the ROWOP operation.

For efficient VLSI implementations, it is desirable to limit interconnect to local connections between PEs to reduce area and delay and to promote a regular layout.

Implementation of an Interpolation Array Processor: An array architecture for a RS (255,239) code with a maximum multiplicity of m=4 will be implemented. The design scales by changing the number of XPROCs and the number of PEs in the XPROCs.

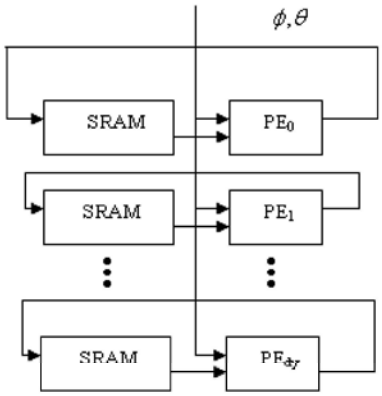

Fig. 3: The polynomial-parallel architecture for polynomial operations.

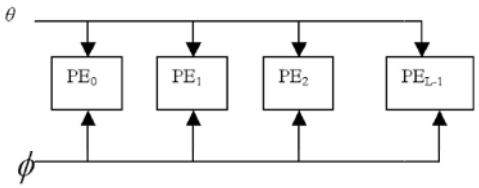

Fig. 4: Monomial parallel architecture for the ROWOP operation.

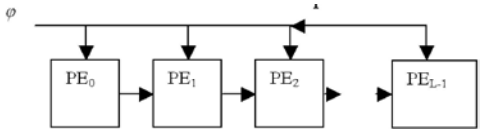

Fig. 5: Monomial parallel architecture for the ZERO operation.

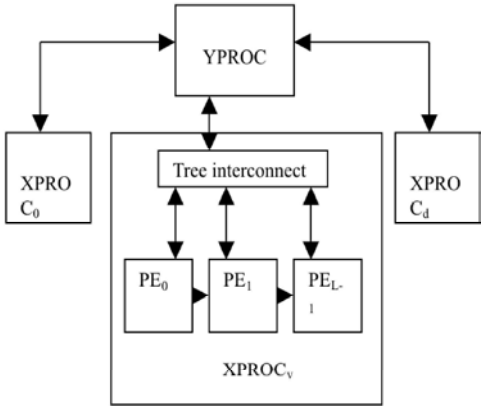

Fig. 6: Architecture for the parallel bivariate polynomial processor.

The design was described in VHDL and was designed to be portable between implementation technologies like FPGA and ASIC technologies. No implementation-dependent optimizations were performed.

Galois Field Arithmetic: Galois field arithmetic has a nice property that addition involves no carriers: the bits in the representation of the field elements are added point wise [9]. For this implementation we use a composite representation of GF(256) that considers an element of GF(2) as two elements of GF(2). Here addition is done with XOR gates also there exists a very efficient multiplier. A GFADD can be implemented with eight XOR gates and has a critical path of one XOR delay. The GFMULT can be implemented with 48 AND gates and 64 XOR gates. The delay is five XOR delays and one AND delays [10].

Top-level Architecture: The top-level architecture consisting of a YPROC, an input buffer and an output buffer is shown below

Data is provided to the processor through 11-bit instructions stored in the input buffer consisting of a 3-bit opcode and an 8-bit data [11-17].

FPGA Implementation: In order to prove the architectural concept, the array processor will be implemented on the FPGA prototyping system.

CONCLUSION

A VLSI architecture is proposed for the implementation of the interpolation step of the Koetter-Vardy algorithm. A parallel array processor architecture proposed is supposed to speed up the interpolation process. The resulting architecture will be well suited for VLSI layout. The array processor will be implemented in FPGA [18-21]. Because of the redecoding, the soft-decision decoder is used for the most unreliable frames which requires least number of cycles to decode. The average throughput of the interpolation processor will be greater.

REFERENCES

1. Warren, J. Gross, Frank R. Kschischang and P. Glenn Gulak, 2007. Architecture and implementation of an interpolation processor for soft-decision Reed-Solomon decoding,"IEEE Trans on VLSI Systems, 15(3).
2. Sundar Raj, M. and T.R. Vasuki, 2013. Automated Anesthesia Controlling System, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1719-1723.

3. Roth, R.M. and G. Ruckenstein, 2000. Efficient decoding of Reed-Solomon codes beyond half the minimum distance, IEEE Trans. Inf. Theory, 46(1): 246-257.
4. Applications of algebraic soft-decision decoding of Reed-Solomon codes, 2006. IEEE Trans. Commun., 54(7): 1224-1234.
5. Koetter, R. and A. Vardy, 2003. Algebraic soft-decision decoding of Reed-Solomon codes," IEEE Trans. Inf. Theory, 49(11): 2809-2825.
6. Gross, W.J., F.R. Kschischang, R. Koetter and P. Gulak, 2005. Towards a VLSI architecture for interpolation-based soft-decision Reed-Solomon decoders, J. VLSI Signal Process., 39: 93-111.
7. Gross, W.J., F.R. Kschischang, R. Koetter and P. Gulak, 2002. A VLSI architecture for interpolation in soft-decision list decoding of Reed-Solomon codes, in Proc. IEEE Workshop on Signal Process. Syst.(SIPS), pp: 39-44.
8. Sundarraj, M., 2013. Study Of Compact Ventilator, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 16(12): 1741-1743.
9. Sundar Raj, M., T. Saravanan and R. Udayakumar, 2013. Energy Conservation Protocol for Real time traffic in wireless sensor networks, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1822-1829.
10. Sundar Raj, M., T. Saravanan and R. Udayakumar, 2013. Data Acquisition System based on CAN bus and ARM, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1857-1860.
11. Zhang, X. and K.K. Parhi, 2005. Fast factorization architecture in soft-decision Reed-Solomon decoding," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 13(4): 413-426.
12. Ahamed, A., R. Koetter and N.R. Shanbhag, 2004, VLSI architectures for soft-decision decoding of Reed-Solomon codes. In Proc. IEEE Int. Conf. Commun., 5: 2584-2590.
13. Gross, W.J., F.R. Kschischang, R. Koetter and P. Gulak, 2002. Simulation results of algebraic soft-decision decoding of Reed-Solomon codes, in Proc.21st Biennial Symp. Commun., pp: 356-360.
14. Guruswami, V. and M. Sudan, 1999. Improved decoding of Reed-Solomon and algebraic geometry codes, IEEE Trans. Inf. Theory, 45(6): 1757-1767.
15. Parhi, K.K., 1999. VLSI Digital Signal Processing Systems. New York: Wiley.
16. Koetter, R., 1996. On algebraic decoding of algebraic-geometric and cyclic codes, Ph.D. dissertation, Dept. Electr. Eng., Linkoping Univ., Linkoping, Sweden.
17. Soljanin, E. and R. Urbanke, 1996. An efficient architecture for implementation of a multiplier and inverter in GF(28), Bell Labs, Murray Hill, NJ, Tech. Rep., BL011217-960308-08TM.
18. Sokeng, S.D., D. Lontsi, P.F. Moundipa, H.B. Jatsa, P. Watcho and P. Kamtchouing, 2007. Hypoglycemic Effect of Anacardium occidentale L. Methanol Extract and Fractions on Streptozotocin-induced Diabetic Rats , Global Journal of Pharmacology, 1(1): 01-05.
19. Prajapati Hetal Ritesh, Brahmkshatriya Pathik Subhashchandra, Vaidya Hitesh Bharatbhai and V. Thakkar Dinesh, 2008. Avian Influenza (Bird Flu) in Humans: Recent Scenario, Global Journal of Pharmacology, 2(1): 01-05.
20. Okafor, P.N., K. Anoruo, A.O. Bonire and E.N. Maduagwu, 2008. The Role of Low-Protein and Cassava-Cyanide Intake in the Aetiology of Tropical Pancreatitis, Global Journal of Pharmacology, 2(1): 06-10.
21. Nahed, M.A., Hassanein, Roba M. Talaat and Mohamed R. Hamed, 2008. Roles of Interleukin-1 (Il-1) and Nitric Oxide (No) in the Anti-Inflammatory Dynamics of Acetylsalicylic Acid Against Carrageenan Induced Paw Oedema in Mice, Global Journal of Pharmacology, 2(1): 11-19.