Proc. of the International Conf. on Electrical, Electronics, Communication, and Information
CECI'2001, March 7-8, Jakarta

Designing 1 bit Error Correcting Circuit on FPGA Using BCH Codes

Sri Suning Kusumawardani*, and Bambang Sutopo†,*IEEE*

* Dept. of Electrical Engineering, Fac. of Engineering, Gadjah Mada University, Jl. Grafika 2, Yogyakarta 55284
Tel. 879505, fax. 879506 email : suning@te.ugm.ac.id
† Dept. of Electrical Engineering, Fac. of Engineering, Gadjah Mada University, Jl. Grafika 2, Yogyakarta 55284
Tel. 879505, fax. 879506 email : bsutopo@te.ugm.ac.id

Abstract– **This paper considers the prototyping of a BCH (*Bose, Chaudhuri, and Hocquenghem)* encoder and decoder using FPGA (*Field Programmable Gate Array*). BCH codes can be defined by two parameters that are code size *n* and the number of errors to be corrected *t*. FPGA is a reprogramable chip. Designing on FPGA is very fast, easy to modify and suitable for prototyping products. This research is our preliminary research on implementation BCH coding in FPGA.**

The results show that the circuits work well, any 1 bit error in any position of 7 bits has been corrected. Our next project is to build 3 bits error correction of 5 bit data, and BCH code size will be 15 bits.

Keywords– BCH codes, encoding, decoding, FPGA, error correcting codes

I. INTRODUCTION

Error Correcting Control is very important in modern communication systems. There are two correcting codes, that are BCH (*Bose, Chaudhuri, and Hocquenghem*) and RS (*Reed-Solomon*) codes, are being widely used in satellite communications, computer networks, magnetic and optic storage systems.

This paper considers the prototyping of a BCH encoder and decoder using FPGA (*Field Programmable Gate Array*). BCH codes operate over finite fields or Galois fields. BCH codes can be defined by two parameters that are code size *n* and the number of errors to be corrected *t*. BCH codes employ sophisticated algorithm and their hardware implementation is rather burdensome. For software implementation is rather slow, consumes more power and less reliable than hardware implementation [1].

FPGA is a reprogramable chip. A design in FPGA can be automatically converted from gate level into layout structure by place and route software. Xilinx Inc. offer a wide range of components, for example, XC4013 offer 13,000 equivalent Nand gates on 546 CLBs (*Configured Logic Blocks*). Designing on FPGA is very fast, easy to modify and suitable for prototyping products, because they are rather expensive and therefore are not economical for mass production [2]. Using ASIC (*Application Specific Integrated Circuit*) implementation might be more appropriate for mass-products, but designing with ASIC is more complex and takes much longer time.

The importance of BCH codes stems from the fact that they are capable of correcting all random patterns of t errors by decoding algorithm that is simple and easily implemented with a reasonable amount of equipment [Rhee].

This research is our preliminary research on implementation BCH coding in FPGA. To simplify the circuit design we have developed one bit correcting circuit. For one bit correction, BCH code need to generate 3 bits parity for 4 bits data, so the length word or code size is 7 bits, this mode usually called (7,4) BCH code.

II. BASIC THEORY

Error control codes rely to a large extent on powerful and elegant algebraic structures called finite fields. A field is essentially a set of elements in which it is possible to add, subtract, multiply and divide field elements and always obtain another element within the set. A finite field is a field containing a finite number of elements.

A field F is a non-empty set of elements with two operators usually called addition and multiplication, denoted "+" and "*" respectively. For F to be a field a number of conditions must hold [Shu Lin] :

1. *Closure*; For every a,b in F
$$c = a + b; d = a * b \quad (1)$$
Where $c, d \in F$.
2. *Associative*; For every a,b,c in F
$$a + (b + c) = (a + b) + c; \quad a * (b * c) = (a * b) * c \quad (2)$$
3. *Identity*; There exists an identity element '*0*' for addition and '*1*' for multiplication that satisfy

$$0 + a = a + 0 = a; and \quad a*1 = 1*a = a \tag{3}$$

for every a in F.

4. *Inverse*; If a is in F, there exist elements b and c in F such that
$$a + b = 0; a*c = 1 \tag{4}$$
Element b is called the additive inverse, $b = (-a)$, element c is called the multiplicative invers, $c = a^{-1}$ $(a \neq 0)$.
5. *Commutative*; For every a,b in F
$$a + b = b + a; a*b = b*a \tag{5}$$
6. *Distributive*; For every a, b, c in F
$$(a + b)*c = a*c + b*c; \tag{6}$$

The existence of a multiplicative invers a^{-1} enables the use of division. This is because for $a,b,c \in F$, $c = b/a$ is defined as $c = b * a^{-1}$. Similarly the existence of an additive inverse $(-a)$ enables the use of subtraction. In this case for $a,b,c \in F$, $c = b - a$ is defined as $c = b + (-a)$.

It can be shown that the set of integers $\{0, 1, 2, ..., p-1\}$ where p is prime, together with modulo p addition and multiplication forms a field. Such a field is called the finite field of order p, or $GF(p)$. In this paper only binary arithmetic is considered, where p is constrained to equal 2. Arithmetic in $GF(2)$ is therefore defined *modulo 2*.

The BCH codes are a class of cyclic codes whose generator polynomial is the product of distinct minimal polynomials corresponding to $\alpha, \alpha^2, ..., \alpha^{2t}$, where $\alpha \in GF(2^m)$ is a root of the primitive polynomial $p(x)$.

An irreducible polynomial $p(x)$ of degree m is said to be primitive if and only if it divides $x^n + 1$ for no n less than $2^m - 1$. In fact, every binary primitive polynomial $p(x)$ of degree m is a factor of $x^{2^m - 1} + 1$. Primitive polynomials of every degree exist over every Galois field, and every Galois field has a primitive element α. Table 1 gives a list of primitive polynomials over $GF(2)$ [Rhee].

Table 1. Primitive polynomials over $GF(2)$

m	$p(x)$	m	$p(x)$
2	$x^2 + x + 1$	7	$x^7 + x^3 + 1$
3	$x^3 + x + 1$	8	$x^8 + x^4 + x^3 + x^2 + 1$
4	$x^4 + x + 1$	9	$x^9 + x^4 + 1$
5	$x^5 + x^2 + 1$	10	$x^{10} + x^3 + 1$
6	$x^6 + x + 1$	11	$x^{11} + x^2 + 1$

Let $m_i(x)$ be the minimal polynomial of α^i . Let $c(x) = c_0 + c_1 x + c_2 x^2 + ... + c_{n-1} x^{n-1}$ be a code polynomial with coefficients from $GF(2)$. If $c(x)$ has $\alpha, \alpha^2, ..., \alpha^{2t}$ as its roots, $c(x)$ is then divisible by the minimal polynomials $m_1(x), m_2(x), ..., m_{2t}(x)$ of $\alpha, \alpha^2, ..., \alpha^{2t}$. The generator polynomial $g(x)$ of the t-error-correcting BCH code of block length $n = 2^m - 1$ and rate k/n is the lowest degree polynomial over $GF(2)$. Thus, the generator polynomial of the code must be the least common multiple of these minimal polynomials. That is,

$$g(x) = LCM\{m_1(x), m_2(x), ..., m_{2t}(x)\} \tag{7}$$

In general, for any positive integers $m \geq 3$ and $t < n/2$, there exists a binary BCH code with parameters of block length $n = 2^m - 1$, number of parity-check bits $n - k \leq mt$, and minimum distance $d_0 = 2t + 1 \leq d_{min}$. The designed distance of the code is $d_0 = 2t + 1$. The minimum distance d_{min} may be larger than d_0. The following steps are used to determine the BCH codes [Rhee].

1. Choose a primitive polynomial of degree m, and construct $GF(2^m)$.
2. Find the minimal polynomial $m_i(x)$ of α^i for $i = 1,2,...,2t$.
3. Obtain $g(x)$.
4. Determine k from $n - k$, which is the degree of $g(x)$.
5. Find the minimum distance $d_{min} \geq 2t + 1$ by referring to the weight of $g(x)$.

Suppose that a code word $c(x)$ is transmitted and that because of the channel error $e(x)$, the received word $r(x)$ is

$$r(x) = c(x) + e(x) \tag{8}$$

$e(x)$ is called the error pattern. No more than t coefficients of $e(x)$ are nonzero. Suppose that ν, $1 \leq \nu \leq t$, errors actually occur and they occur in unknown locations $j_1, j_2, ..., j_\nu$, that is,

$$e(x) = \sum_{\lambda=1}^{o} x^{j\lambda}, 0 \leq j_\lambda \leq n-1 \tag{9}$$

Since $\alpha, \alpha^2, ..., \alpha^{2t}$ are roots of each code polynomial, $c(\alpha^i)=0$, for $1 \leq i \leq 2t$. Therefore, from Equation (8), it follows that

$$r(\alpha^i) = e(\alpha^i), i = 1,2,...,2t \tag{10}$$

The decoding of a received BCH word requires that three successive computational processed performed over $GF(2^m)$ be executed. These processes are the syndrome computations, error-locator polynomial determination, and the Chien search (with error-value computational for nonbinary codes).

Syndrome Computations

The first step in decoding a t-error-correction BCH code is to compute the $2t$ syndrome components $s_1, s_2, ..., s_{2t}$. These syndrome components may be obtained by substituting the field elements $\alpha, \alpha^2, ..., \alpha^{2t}$ into the received polynomial $r(x)$. Thus, the ith component of the syndrome is [Shu Lin]

$$\begin{aligned} s_i &= r(\alpha^i) \\ s_i &= r_{n-1}(\alpha^i)^{n-1} + r_{n-2}(\alpha^i)^{n-2} + ... + r_1\alpha^i + r_0 \\ s_i &= (...((r_{n-1}\alpha^i + r_{n-2})\alpha^i + r_{n-3})\alpha^i + ... + r_1)\alpha^i + r_0 \\ 1 &\leq i \leq 2t \end{aligned} \tag{11}$$

The syndrome components are a function of the field elements of $GF(2^m)$. Thus, each syndrome component is

computed by dividing $r(x)$ by the minimal polynomial $m_i(x)$, $1 \le i \le 2t$, of α^j such that

$$r(x) = q_i(x)m_i(x) + \gamma_i(x)\} \quad (12)$$

The remainder $\gamma_i(x)$, where $x = \alpha^j$, is the syndrome component s_i since $m_i(\alpha^j) = 0$. Thus, in general, computing $r(\alpha^j)$ is equivalent to computing $\gamma_i(\alpha^j)$. Hence, if it is combined with Equation (10), the syndrome component is expressed as

$$s_i = \gamma_i(\alpha^i) = r(\alpha^i) = e(\alpha^i), i = 1,2,...,2t \quad (13)$$

from which we see that the syndrome s depends only on the error pattern e. It thus follows that we have a set of equations that relate the syndrome components and unknown parameters (the error-location numbers) $\alpha^{j\lambda}$, $1 \le \lambda \le \upsilon$.

$$s_i = \sum_{\lambda=1}^{\upsilon} (\alpha^{j\lambda})^i, 1 \le i \le 2t \quad (14)$$

Consequently, the decoding algorithm of the BCH codes is the way to solve these power sum symmetric functions (Equation 14) and to find the unknown numbers $\alpha^{j\lambda}$, $1 \le \lambda \le \upsilon$, from the syndrome components s_i.

The Error-Locator Polynomial

Suppose that $\upsilon \le t$ errors actually occur. The error-locator polynomial $\sigma(x)$ be

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + ... + \sigma_\upsilon x^\upsilon$$
$$\sigma(x) = (1 + \beta_1 x)(1 + \beta_2 x)...(1 + \beta_\upsilon x) \quad (15)$$

by letting $\beta_\lambda = \alpha^{j\lambda}$ for simplicity. Its coefficients and the error-location numbers are related by the following set of equations [Rhee] :

$$\sigma_0 = 1$$
$$\sigma_1 = \beta_1 + \beta_2 + ... + \beta_\upsilon$$
$$\sigma_2 = \beta_1\beta_2 + \beta_2\beta_3 + ... + \beta_{\upsilon-1}\beta_\upsilon \quad (16)$$
$$...$$
$$\sigma_\upsilon = \beta_1\beta_2...\beta_\upsilon$$

The σ_i, $0 \le i \le \upsilon$, are closely related to the syndrome components s_j, $1 \le j \le \upsilon+1$ [Rhee]. The algorithm for finding $\sigma(x)$ for the error correction of $t = 1$ random error is summarized as follows [Rhee].

1. $\sigma(x) = 1$, $s_1 = s_3 = 0$ for no error.
2. $\sigma(x) = 1 + s_1 x$, $s_1 \neq 0$, $s_3 = s_1^3$, for a lone error.

III. EXPERIMENTAL RESULTS

Consider the single-error-correcting (7,4) BCH code. Let α be a primitive element of the Galois field $GF(2^3)$ such that $1 + \alpha + \alpha^3 = 0$. If $m_i(x)$, $i = 1,2, ..., 6$, denote the minimal polynomials of α^j, which are the elements of $GF(2^3)$, we then have the list given by Table 2.

The generator polynomial of the (7,4) BCH code can be given by

$$g(x) = LCM\{m_1(x), m_2(x)\}$$
$$g(x) = m_1(x)$$
$$g(x) = x^3 + x + 1$$

Table 2. Minimal polynomials of the elements in $GF(2^3)$

Elements	Conjugates	Minimal polynomials
α	α^2, α^4	$x^3 + x + 1$
α^2	α^4, $\alpha^8 = \alpha$	$x^3 + x + 1$
α^3	α^6, $\alpha^{12} = \alpha^5$	$x^3 + x^2 + x + 1$
α^4	$\alpha^8 = \alpha$, $\alpha^{16} = \alpha^2$	$x^3 + x + 1$
α^5	$\alpha^{10} = \alpha^3$, $\alpha^{20} = \alpha^6$	$x^3 + x^2 + x + 1$
α^6	$\alpha^{12} = \alpha^5$, $\alpha^{24} = \alpha^3$	$x^3 + x^2 + x + 1$

Figure 1 shows the encoder [Rhee].

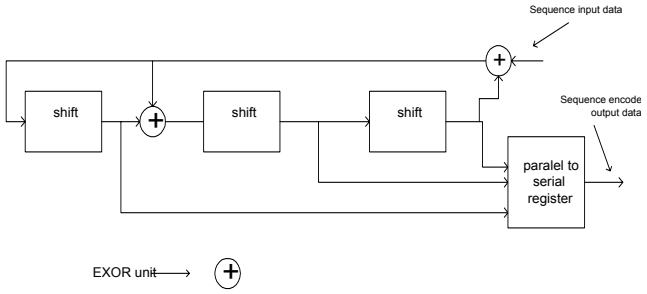

Fig.1. BCH Encoding logic algorithm

Based on Figure 1, the encoding circuit for the (7,4) BCH code is easily implemented as shown in Figure 2. The output of figure 2 is a serial bit of 7 bit data generated by BCH encoder. The input data is 4 bits.

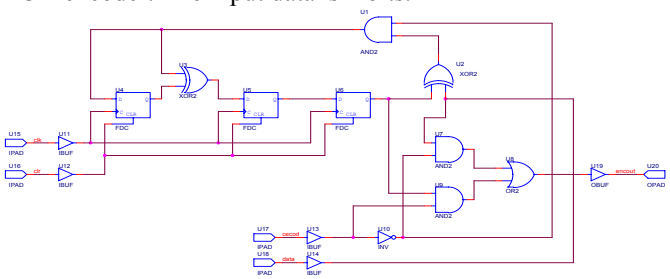

Fig.2. BCH Encoding Logic Circuit implemented in FPGA

Let $r(x)$ be the received polynomial. To compute the syndrome digits, we divide $r(x)$ by $m_1(x)$, and the remainder $\gamma(x)$ is assumed to be $\gamma(x) = \gamma_0 + \gamma_1 x + \gamma_2 x^2$. Substituting α and α^2 into $\gamma(x)$, we obtain

$$\gamma(\alpha) = s_1 = \gamma_0 + \gamma_1\alpha + \gamma_2\alpha^2$$
$$\gamma(\alpha^2) = s_2 = \gamma_0 + \gamma_1\alpha^2 + \gamma_2\alpha^4$$
$$= \gamma_0 + \gamma_2\alpha + (\gamma_1 + \gamma_2)\alpha^2, \text{respectively}.$$

For a single-bit correction the syndrome logic algorithm is shown in figure 3 and the logic circuit is in figure 4.

The error-locator polynomial:

$$\sigma(x) = \sigma_0 + \sigma_1 x = 1 + \beta_1 x = 1 + s_1 x$$

Only when $s_1 \neq 0$ and $s_3 = s_1^3$.

Finally, if no error in *r(x)* exist, the decoder generates no syndromes. Therefore, the error-locator polynomial simply becomes $\sigma(x) = 1$. It can be done in hardware using Chien's searching unit shown in Figure 5.Implementation on FPGA of Figure 5 shown in Figure 6.

Circuit design and simulation has been done in OrCAD Version 9.1, before it is implemented in FPGA Xilinx XC4013. The result show that the circuits work well, any 1 bit error in any position of 7 bits has been corrected. Our next project is to build 3 bits error correction of 5 bit data, and BCH code size will be 15 bits. To justify the work we have done, we attached the encoding and decoding circuit of BCH codes. This circuit comprises of three parts, which are BCH encoding unit, Syndrome Unit, and Chien's error-location searching unit.

IV. CONCLUSIONS

The circuits work well, any 1 bit error in any position of 7 bits has been corrected. Our next project is to build 3 bits error correction of 5 bit data, and BCH code size will be 15 bits. To justify the work we have done, we attached the encoding and decoding circuit of BCH codes. This circuit comprises of three parts, which are BCH encoding unit, Syndrome Unit, and Chien's error-location searching unit.

REFERENCES

[1] B. Sutopo, "Designing 4 point Winograd small FFT on FPGA", Quality in Research Seminar, University of Indonesia, 2000.

[2] E. Jamro, "The Design of VHDL Based Synthesis Tool for BCH Codecs", M.Phil Thesis, School of Engineering, The University of Huddersfield, 1997.

[3] M.Y. Rhee, "Error Correcting Coding Theory", McGraw-Hill, Singapore, 1989.

[4] S. Lin, and D.J. Costello, Jr., "Error Control Coding", Prentice-Hall, New Jersey, 1983.

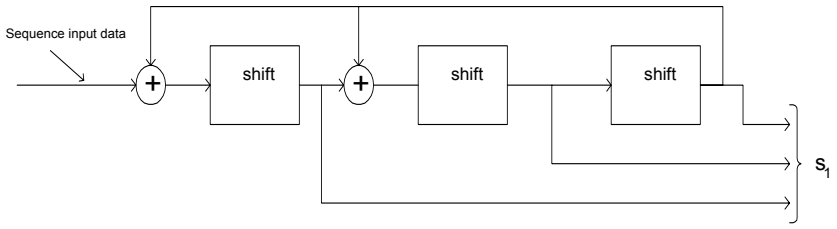

Fig.3. Syndrome Unit

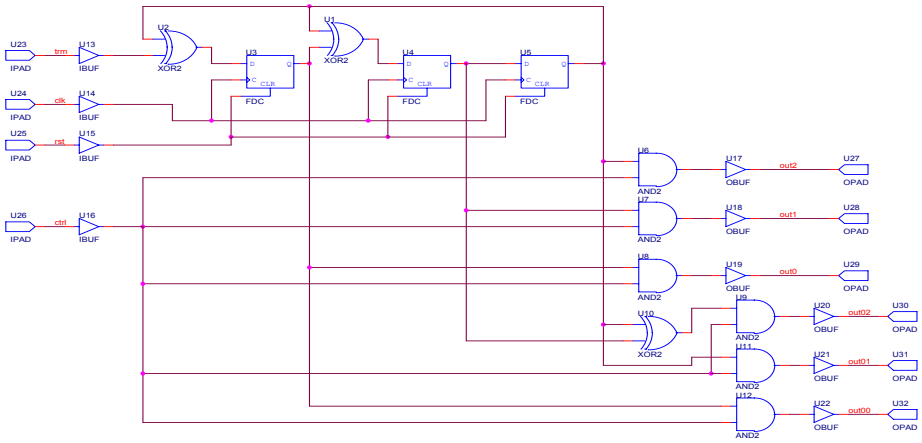

Fig.4. Syndrome Logic Circuit in FPGA

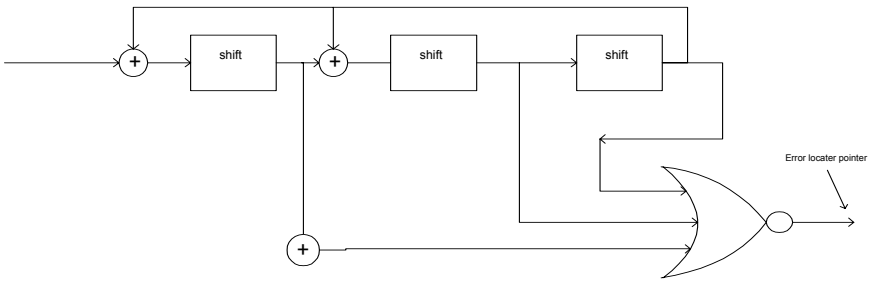

Fig.5. Chien's Error location searching Unit

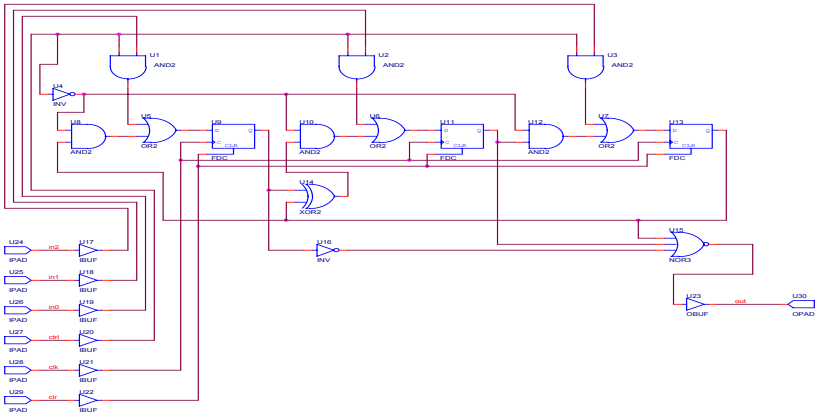

Fig.6. Implementation on FPGA of figure 5