Journal of Informatics and Mathematical Sciences
Vol. 7, No. 1, pp. 1–11, 2015
ISSN 0975-5748 (online); 0974-875X (print)
Published by RGN Publications
http://www.rgnpublications.com

An Investigation of Quaternary $[5,3]$ Error Correcting Codes and their Implementation with Binary Devices

Research Article

A.K.M. Toyarak Rian[1], Partha Pratim Dey[2,*], Farzana Karim Elora[1]

[1] *Department of Electrical and Computer Engineering, North South University, Bangladesh*
[2] *Department of Mathematics and Physics, North South University, Bangladesh*
***Corresponding author:** ppd@northsouth.edu

Abstract. In this paper we investigate the existence, equivalence and some other features of quaternary $[5,3]$ error correcting codes. We also discuss the binary hardware devices that could be used to code and decode messages when these $[5,3]$ error-correcting codes are used.

Keywords. Generalized linear code; Generator matrix; Equivalent code; Gates

MSC. 53C15; 53C25

Received: November 7, 2014 **Accepted:** December 15, 2014

1. Introduction

Let F be the GF(q), the Galois field with q elements. An $[n,k]$ linear code over GF(q) is a k-dimensional subspace of F^n, the space of all n-tuples with components from F. Since a linear code is a vector sub-space, it can be given by a basis. A *generator matrix* is a $k \times n$ dimensional matrix whose rows are the basis vectors of the code. For an acquaintance with coding theory at a basic level, the reader may please consult [1, 2, 3].

A very important concept in coding is the weight of a vector v. By definition, this is the number of non-zero components v has and is denoted by $wt(v)$. The minimum weight of a code, denoted by d is the weight of a non-zero vector of smallest weight in the code. A well-known theorem says that if d is the minimum weight of a code C, then C can correct $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ or fewer errors, and conversely. An $[n,k]$ linear code with minimum distance d is often called an $[n,k,d]$ code. A quaternary code is a $[n,k,d]$ code over GF(4). In this paper, we intend to explore the

[5,3,3] linear codes over GF(4). Recall that GF(4) denotes the Galois field of order 4 comprising of $0,1,\omega$ and ϖ with the following addition and multiplication tables:

$+$	0	1	ω	ϖ
0	0	1	ω	ϖ
1	1	0	ϖ	ω
ω	ω	ϖ	0	1
ϖ	ϖ	ω	1	0

$\times$	0	1	ω	ϖ
0	0	0	0	0
1	0	1	ω	ϖ
ω	0	ω	ϖ	1
ϖ	0	ϖ	1	ω

2. Existence of a $[5,3,3]$ Linear Code

By singleton bound $d \le n-k+1$ for an $[n,k,d]$ code. Hence for a [5,3] code, $d=3$ is the maximum attainable minimum distance. On the other hand, to be 1 error correcting, the minimum distance of a linear code should be at least 3. Hence, an 1 error correcting [5,3] code, if it exists, has to be a [5,3,3] code. In this paper, we will show that there exist no [5,3,3] code over fields of order 2 and order 3, but there do exist [5,3,3] codes over fields of order 4 or greater.

Theorem 2.1. *There exists no* [5,3] *one error correcting binary code.*

Proof. Let M be a generator matrix of a [5,3] binary code. Then

$$M=\begin{bmatrix}1&0&0&a_{11}&a_{12}\\0&1&0&a_{21}&a_{22}\\0&0&1&a_{31}&a_{32}\end{bmatrix}$$

where $a_{ij}\in \mathrm{GF}(2)$ for each i and j, $1\le i\le 3$, $1\le j\le 2$.

If the code is to be error correcting, the minimum distance d should be at least 3. Hence $a_{ij}\ne 0$ for each i and j, $1\le i\le 3$, $1\le j\le 2$. One then obtains the following equivalence diagram where r_i and c_i denote the ith row and ith column respectively.

$$M=\begin{bmatrix}1&0&0&a_{11}&a_{12}\\0&1&0&a_{21}&a_{22}\\0&0&1&a_{31}&a_{32}\end{bmatrix}\xrightarrow{a_{11}^{-1}r_1,a_{21}^{-1}r_2,a_{31}^{-1}r_3}\begin{bmatrix}a_{11}^{-1}&0&0&1&a_{11}^{-1}a_{12}\\0&a_{21}^{-1}&0&1&a_{21}^{-1}a_{13}\\0&0&a_{31}^{-1}&1&a_{31}^{-1}a_{14}\end{bmatrix}$$

$$\xrightarrow{a_{11}c_1,a_{21}c_2,a_{31}c_3}\begin{bmatrix}1&0&0&1&a_{11}^{-1}a_{12}\\0&1&0&1&a_{21}^{-1}a_{13}\\0&0&1&1&a_{31}^{-1}a_{14}\end{bmatrix}\xrightarrow{a=a_{11}^{-1}a_{12},b=a_{21}^{-1}a_{13},c=a_{31}^{-1}a_{14}}\begin{bmatrix}1&0&0&1&a\\0&1&0&1&b\\0&0&1&1&c\end{bmatrix}$$

$$\xrightarrow{a^{-1}c_5}\begin{bmatrix}1&0&0&1&1\\0&1&0&1&a^{-1}b\\0&0&1&1&a^{-1}c\end{bmatrix}\xrightarrow{x=a^{-1}b,y=a^{-1}c}\begin{bmatrix}1&0&0&1&1\\0&1&0&1&x\\0&0&1&1&y\end{bmatrix}=G.$$

Since code is binary, and each of x and y is nonzero, $x=y=1$. Hence

$$G=\begin{bmatrix}1&0&0&1&1\\0&1&0&1&1\\0&0&1&1&1\end{bmatrix}.$$

If we now add the 1st and 2nd row vectors of G, we get the codeword $(1,1,0,0,0)$ which has weight 2. There therefore exists no error correcting binary [5,3] code. □

Theorem 2.2. *There exists no* [5,3] *one error correcting ternary code.*

Proof. Let M be a generator matrix of a [5,3] binary code. Then

$$M = \begin{bmatrix} 1 & 0 & 0 & a_{11} & a_{12} \\ 0 & 1 & 0 & a_{21} & a_{13} \\ 0 & 0 & 1 & a_{31} & a_{14} \end{bmatrix}.$$

where $a_{ij} \in \mathrm{GF}(3)$ for each i and j, $1 \le i \le 3$, $1 \le j \le 2$.

Using the equivalence diagram as in Theorem 2.1 above, we get that M is equivalent to

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & x \\ 0 & 0 & 1 & 1 & y \end{bmatrix}$$

where $xy = 11, 12, 21$ or 22. Thus, we obtain 4 generator matrices from G:

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \qquad G_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 \end{bmatrix},$$

$$G_3 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \qquad G_4 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 2 \end{bmatrix}.$$

It is an easy exercise to check that each of G_i above generates a code, which has a codeword of weight 2. There exists no [5,3] one error correcting ternary code. □

The following theorem shows that there do exist 1-error correcting quaternary [5,3] codes and they are all equivalent.

Theorem 2.3. *An* 1*-error correcting* [5,3] *quaternary code is equivalent to the code with the following generator matrix* $\overline{G}$ *where*

$$\overline{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega \\ 0 & 0 & 1 & 1 & \varpi \end{bmatrix}$$

and 1 *and* ω *and* ϖ *are nonzero elements of* GF(4) *with* 1 *representing the identity element.*

Proof. Let M be a generator matrix of a 1-error correcting [5,3] quaternary code. Then as in Theorem 2.1 above, M can be shown to be equivalent to

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & x \\ 0 & 0 & 1 & 1 & y \end{bmatrix}$$

where x and y are nonzero elements of GF(4). Notice that x cannot be 1, as then the first two rows of G if added will produce a codeword of weight 2. On the other hand x and y cannot be same, as then the last two rows of G if added will give a codeword of weight 2. Hence, we obtain only two codes from G with the following generator matrices:

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega \\ 0 & 0 & 1 & 1 & \varpi \end{bmatrix} \quad \text{or} \quad G_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \varpi \\ 0 & 0 & 1 & 1 & \omega \end{bmatrix}$$

which could have error correction power.

We now see in the diagram below that G_1 and G_2 generate equivalent codes.

$$G_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega \\ 0 & 0 & 1 & 1 & \varpi \end{bmatrix} \xrightarrow{swap(r_1,r_2)} \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \varpi \\ 0 & 0 & 1 & 1 & \omega \end{bmatrix} \xrightarrow{swap(c_1,c_2)} \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & \varpi \\ 0 & 0 & 1 & 1 & \omega \end{bmatrix} = G_2 .$$

Notice that $\overline{G} = G_1$.

We now show that the code generated by $\overline{G}$ is able to correct 1 error. Notice that the parity check matrix of $\overline{G}$ is

$$\overline{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & \omega & \varpi & 0 & 1 \end{bmatrix}$$

and no two columns of $\overline{H}$ are dependent. However, there exist 3 columns of $\overline{H}$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

which are dependent. Then by a well-known theorem [1], the minimum weight of the code generated by $\overline{G}$ is 3. □

The next theorem shows that there do always exist an 1-error correcting [5,3] code over GF(q).

Theorem 2.4. *Let* GF(q) *be a field of order q where $q \geq 4$, then there do always exist a* [5,3,3] *code over* GF(q).

Proof. Let M be a generator matrix of a [5,3] code over GF(q), $q \geq 4$. Then

$$M = \begin{bmatrix} 1 & 0 & 0 & a_{11} & a_{12} \\ 0 & 1 & 0 & a_{21} & a_{13} \\ 0 & 0 & 1 & a_{31} & a_{14} \end{bmatrix}$$

where $a_{ij} \in \text{GF}(q)$ for each i and j, $1 \leq i \leq 3$, $1 \leq j \leq 2$.

Using the equivalence diagram as in Theorem 1 above, we get that M is equivalent to

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & x \\ 0 & 0 & 1 & 1 & y \end{bmatrix}$$

Since $q \geq 4$, exist nonzero $x, y \in \text{GF}(q)$ such that 1, x and y are all distinct. Then no two columns of

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & x & y & 0 & 1 \end{bmatrix}$$

are dependent and exist 3 columns of H

$$\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix} \text{ and } \begin{bmatrix}0\\1\end{bmatrix}$$

which are dependent. Hence the minimum weight of the code generated by G or M is 3. □

Now a word or two about the weight distribution of the 1-error correcting [5,3] quaternary codes. We have written a MAPLE program to compute the weight distribution of the code generated by $\overline{G} = \begin{bmatrix}1 & 0 & 0 & 1 & 1\\0 & 1 & 0 & 1 & \omega\\0 & 0 & 1 & 1 & \varpi\end{bmatrix}$ and obtained the following result: $A_0 = 1$, $A_3 = 30$, $A_4 = 15$ and $A_4 = 18$, where A_i indicates the number of codewords of weight i.

3. Error-Correcting Devices and the Related Mathematics

An ordered pair of field elements of GF(2) is a Cartesian pair $(a,b) \in \mathrm{GF}(2) \times \mathrm{GF}(2)$, denoted below by ab for the sake of convenience. Obviously there are 4 such pairs, namely 00, 01, 10 and 11. We represent the field elements of GF(4) by an ordered pair of field elements of GF(2) as follows:

$$0 \to 00,\ 1 \to 11,\ \omega \to 01 \text{ and } \varpi \to 10.$$

Then the addition and multiplication tables of GF(4), using this representation, assume the following form:

+	00	11	01	10
00	10	11	01	10
11	11	00	10	01
01	01	10	00	11
10	10	01	11	00

×	00	11	01	10
00	00	00	00	00
11	00	11	01	10
01	00	01	10	11
10	00	10	11	01

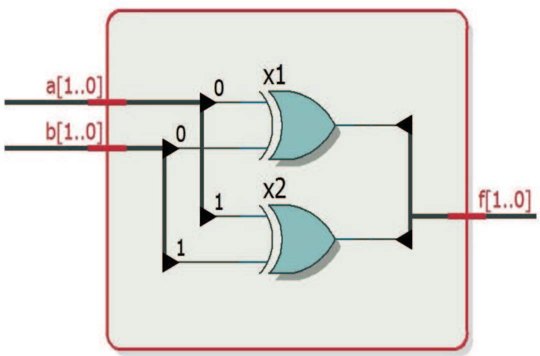
One can easily check that if $a_1a_0 + b_1b_0 = f_1f_0$, then

$$f_0 = (a_0 \oplus b_0)\cdot\overline{(a_1 \oplus b_1)} + (a_0 \oplus b_0)\cdot(a_1 \oplus b_1)$$

and

$$f_1 = \overline{(a_0 + b_0)}\cdot(a_1 + b_1) + (a_0 + b_0)\cdot(a_1 \oplus b_1).$$

Given below is a gate implementation of the addition table of GF(4) that we will call adder and denote by *add* 2.



Note that though *add* 2 adds only 2 elements of GF(2), using recursion, we can build add *add i* for any number of *i* elements from GF(4).

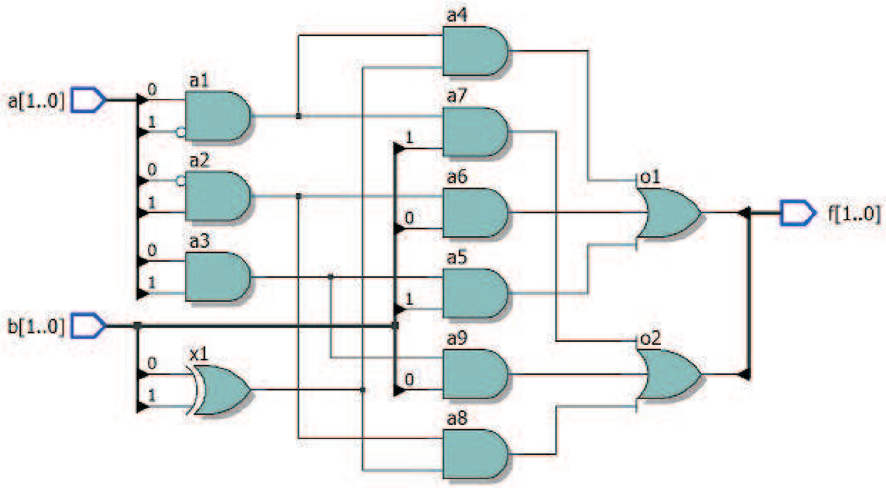
Similarly, if $a_1a_0 \cdot b_1b_0 = f_1f_0$, then

$$f_0 = \overline{a_1}a_0b_1 + a_1\overline{a_0}(b_1 \oplus b_0) + a_1a_0b_0$$

and

$$f_1 = \overline{a_1}a_0(b_1 \oplus b_0) + a_1a_0b_1 + a_1a_0b_1 + a_1\overline{a_0}b_0.$$

Given below is a binary logic gate level implementation of the multiplication table of GF(4) that we will call 'multiplier 'denoted by *mul*.



We will use the adders and multiplier circuits that we designed above to add and multiply the field elements of GF(4) as we construct below the design of the encoder. Let C be the code generated by G where

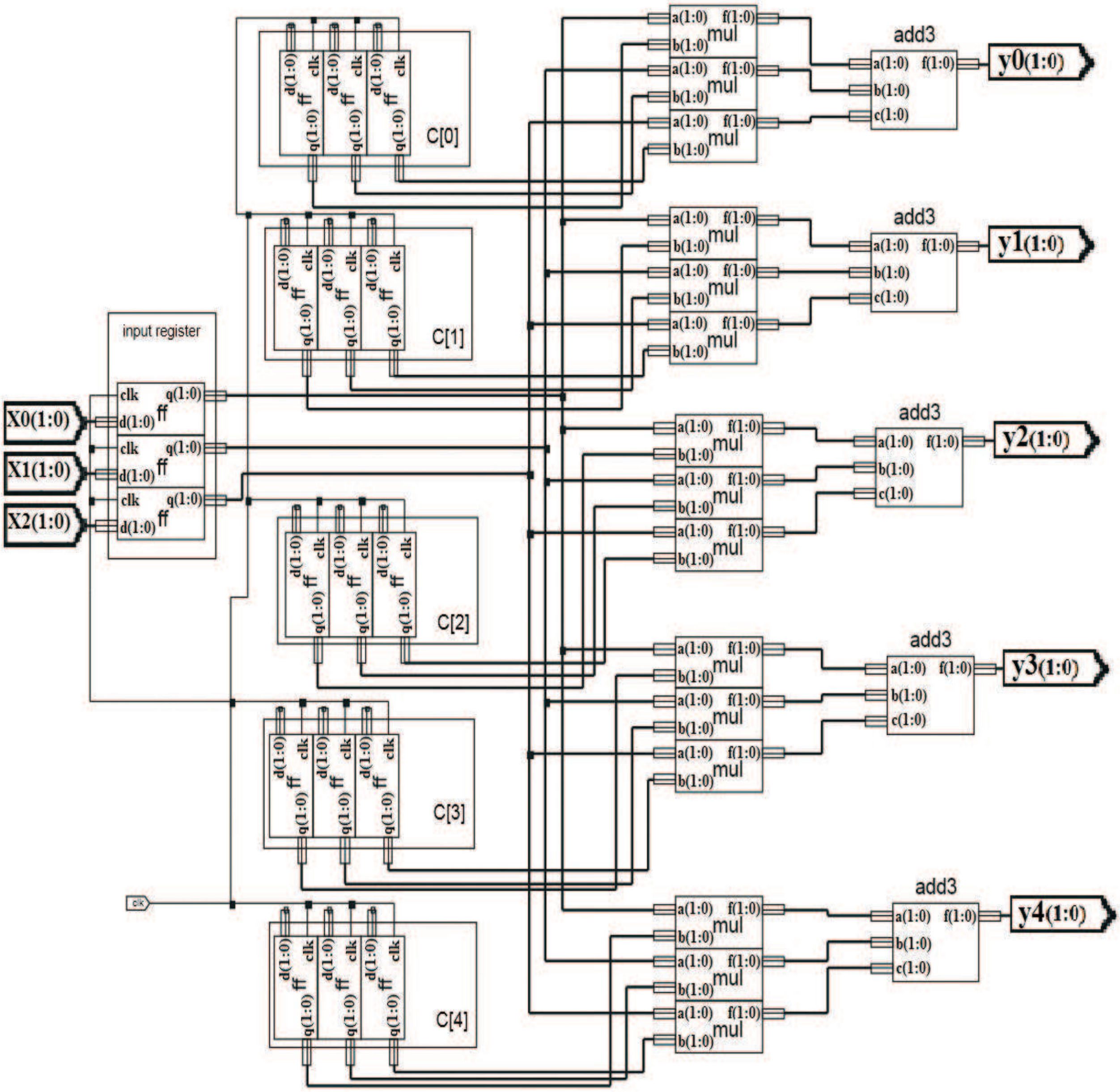
$$G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} \end{bmatrix}.$$

Then $C = \{x_0r_0 + x_1r_1 + x_2r_2 \mid x_0, x_1, x_2 \in \text{GF}(4)\}$ where r_i indicates the ith row of G. The vector $m = (x_0, x_1, x_2)$ is called the message and $c = (y_0, y_1, y_2, y_3, y_4)$ is called the codeword.

We compute the 5 coordinates of the codeword using the following set of 5 equations:

$$\left.\begin{aligned} y_0 &= x_0g_{11} + x_1g_{21} + x_2g_{31} \\ y_1 &= x_0g_{12} + x_1g_{22} + x_2g_{32} \\ y_2 &= x_0g_{13} + x_1g_{23} + x_2g_{33} \\ y_3 &= x_0g_{14} + x_1g_{24} + x_2g_{34} \\ y_4 &= x_0g_{15} + x_1g_{25} + x_2g_{35} \end{aligned}\right\} \qquad \text{(I)}$$

Hence, an encoder transforms the message $m = (x_0, x_1, x_2)$ into a codeword $c = (y_0, y_1, y_2, y_3, y_4)$ using the generator matrix G. This process of transforming a message m into a codeword c is called an encoding and the device that carries out this transformation is called an encoder. Given below is a circuit diagram that illustrates the encoding process of a message.



It stores three xs of message $m = (x_0, x_1, x_2)$ into three flip-flops of input register. It also stores the ith column $[g_{1i}, g_{2i}, g_{3i}]^{transpose}$ of the generator matrix G in the register $C[i]$ for each $i = 0, 1, \ldots, 4$. Note that there are 5 circuits, each of which comprises of 3 *mul* circuits. Of these five, the ith circuit from top, computes $x_0 g_{1i}$ in the top *mul*, $x_1 g_{1i}$ in the middle *mul* and $x_2 g_{3i}$ in the bottom *mul*. The *add* i then adds these $x_0 g_{1i}$, $x_1 g_{2i}$ and $x_2 g_{3i}$, and yields the ith coordinate y_i of the codeword c.

Next, we discuss a device, known as syndrome calculator. We begin by explaining the mathematics involved. Let

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \end{bmatrix}$$

be a parity check matrix of the code C. Suppose a codeword $c = (y_0, y_1, y_2, y_3, y_4)$ is send through the transmission channel and the vector $r = (r_0, r_1, r_2, r_3, r_4)$ has been received.

Then the syndrome of r, denoted by $s(r) = (s_0, s_1)$, is given by:

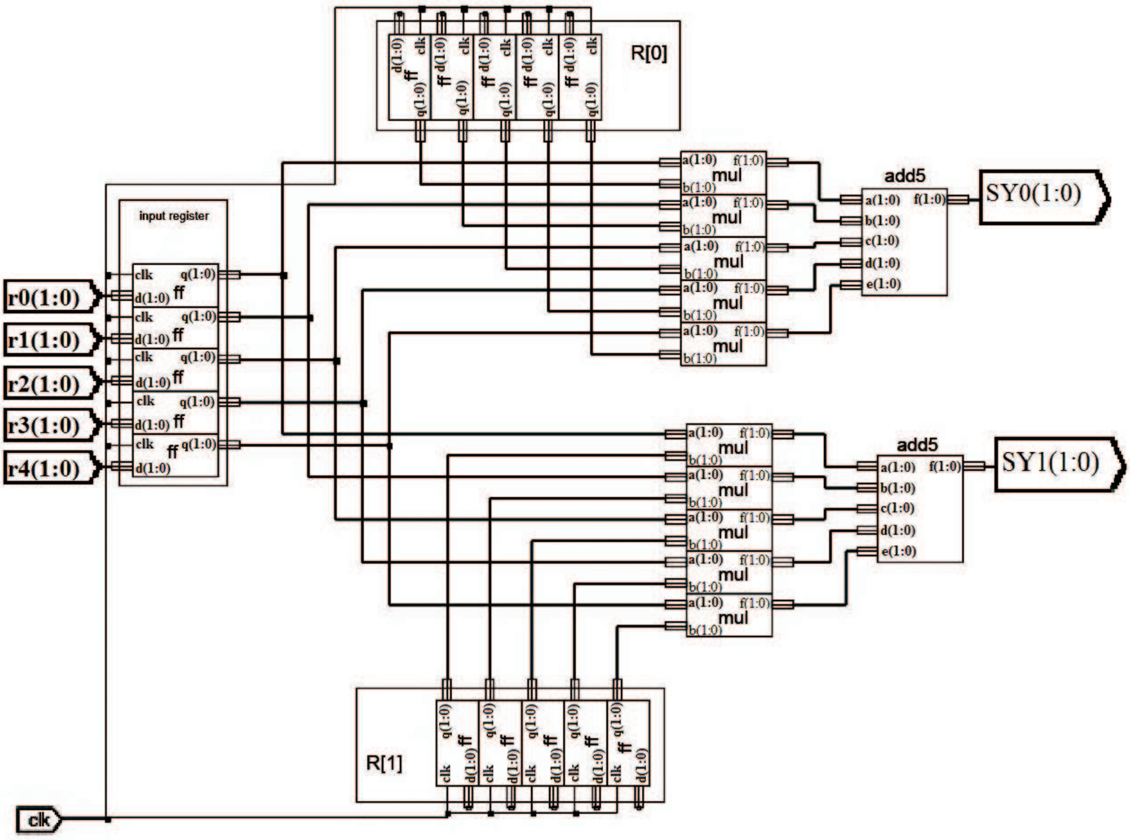
$$\begin{bmatrix} s_0 \\ s_1 \end{bmatrix} = Hr' = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix}$$

$$= \begin{bmatrix} r_0h_{11} + r_1h_{12} + r_2h_{13} + r_3h_{14} + r_4h_{15} \\ r_0h_{21} + r_1h_{22} + r_2h_{23} + r_3h_{24} + r_4h_{25} \end{bmatrix}.$$

Hence, we obtain the following pair of equations:

$$\left. \begin{array}{l} r_0h_{11} + r_1h_{12} + r_2h_{13} + r_3h_{14} + r_4h_{15} \\ r_0h_{21} + r_1h_{22} + r_2h_{23} + r_3h_{24} + r_4h_{25} \end{array} \right\} \qquad \text{(II)}$$

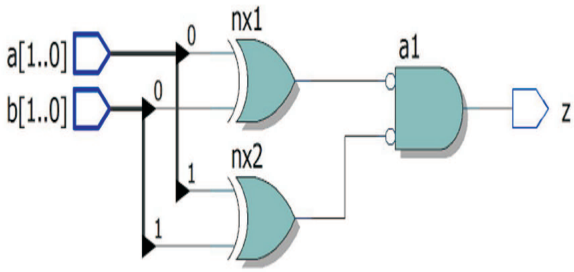
The device below computes $s(r) = (s_0, s_1)$.



It stores the five r_is of r in the five flip-flops of input register. It also stores the five elements $h_{11}, h_{12}, h_{13}, h_{14}, h_{15} \in \text{GF}(4)$ of the first row of the parity check matrix H in the five flip-flops of the register $R[0]$. The five elements $h_{21}, h_{22}, h_{23}, h_{24}, h_{25} \in \text{GF}(4)$ of the second row of H are stored in $R[1]$, which like $R[0]$ comprises of five flip-flops. Notice that there are two blocks, each comprising of five gates, in the right part of the device. The upper block computes $r_0h_{11}, r_1h_{12}, r_2h_{13}, r_3h_{14}, r_4h_{15}$. These terms are then added by add 5 to produce output s_0.

Similarly, the lower block computes $r_0h_{21}, r_1h_{22}, r_2h_{23}, r_3h_{24}, r_4h_{25}$, which are then added by add 5 to output s_1.

Next, we discuss comparator. A comparator is a device that takes two elements x and y of GF(4) as inputs and yields 1 if the elements are same and 0 otherwise. If viewed as an ordered pair, $x = a_1a_0$ and $y = b_1b_0$, then it is an easy exercise to see that the output z of the comparator is given by $z = \overline{(a_0 + b_0)} \cdot \overline{(a_1 + b_1)}$. Given below is a logic design that implements a comparator. We will need this device when we build the decoder.



Finally, we discuss the decoder. The decoder detects and corrects the error (if any) in the received vector r and outputs the codeword c that was sent through the channel. Suppose we send the codeword c. However, due to noise in the transmission channel, an error occurs and the received vector r is different by a single bit from c. Then $e = r - c$ is called error vector and has the shape $e = (0, \dots, 0, \alpha, 0, \dots, 0)$, where $\alpha \neq 0$ and the coordinate position of α indicates the coordinate position of the bit that was corrupted. To recover the codeword $c$from the received vector r we first form the parity check matrix H using the basis vectors of ker(G), where G is the generator matrix of code C.

We compute Hr^t as follows:

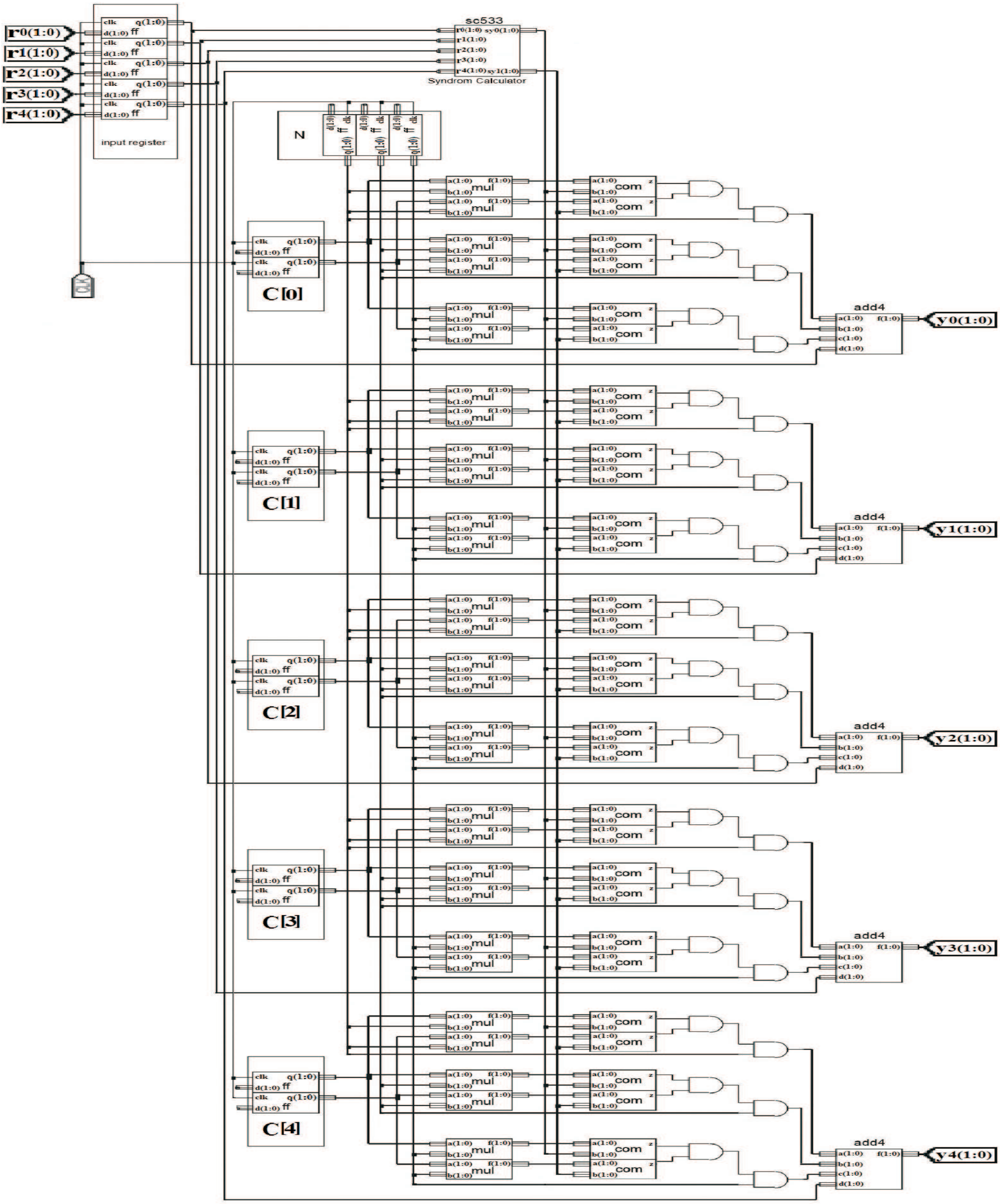
$$\begin{aligned} Hr^t &= H(c + (r - c))^t \\ &= H(c + e)^t \\ &= Hc^t + He^t. \end{aligned}$$

Since $c \in \ker H$, $Hc^t = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.

Since $e = (0, \dots, 0, \alpha, 0, \dots, 0)$, $He^t = \alpha \cdot$ a column of H. Hence $Hr^t = \alpha \cdot$ a column of $Hr^t = \alpha \cdot j$th. If H column of H where $j \in \{1,2,3,4,5\}$ and $\alpha \in$ GF(4) such that $\alpha \neq 0$, then the error has occurred in the jth bit of the sent code-word and the error vector e has field element α in its jth coordinate position and zeros in others i.e. $e = (0, 0, \dots, \alpha, \dots, 0, 0)$ where α is the jth bit of e. If Hr^t is zero vector, no error has occurred during transmission. We compute $r - e$ to recover the codeword c.

Given in the next page is a circuit design of the decoder. The received vector $r = (r_0, r_1, r_2, r_3, r_4)$ is stored in the five flip-flops of the input register. The register N comprises of three flip-flops which contain 1, ω and φ. On the other hand, register $C[i]$ comprises if 2 flip-flops, which contains the 2 elements of the ith column $[h_{1i}\ h_{2i}]^{transposed}$ of the parity check matrix H. Notice that there are 3 pairs of mul circuits adjacent to and at a level of$C[i]$. The top pair multiplies $[h_{1i}\ h_{2i}]^{transposed}$ with 1, the middle pair with ω and the bottom pair with ϖ

respectively. Now notice that there are also 3 pairs of comparators next to these *mul* circuits at their level. The top comparator pair compares $1\cdot[h_{1i}\ h_{2i}]^{transposed}$ with $[s_0\ s_1]^{transposed}$, the middle pair compares $\omega\cdot[h_{1i}\ h_{2i}]^{transposed}$ with $[s_0\ s_1]^{transposed}$ and the bottom pair compares $\varpi\cdot[h_{1i}\ h_{2i}]^{transposed}$ with $[s_0\ s_1]^{transposed}$.



Three cases arise here:

(1) If $1 \cdot [h_{1i}\ h_{2i}]^{transposed} = [s_0\ s_1]^{transposed}$, then 1 enters into the *add* 4 from the topmost pair of "and" gates, whereas from the other two pairs of the "and" gates 2 zeros enter into *add* 4. The *add* 4 then adds $1, 0, 0, r_i$ and yields the ith coordinate of the transmitted codeword c.

(2) If $\omega \cdot [h_{1i}\ h_{2i}]^{transposed} = [s_0\ s_1]^{transposed}$, then ω enters into the *add* 4 from the middle pair of "and" gates, whereas from the other two pairs of the "and" gates 2 zeros enter into *add* 4. The *add* 4 then adds $\omega, 0, 0, r_i$ and yields the ith coordinate of the transmitted codeword c.

(3) Finally if $\varpi \cdot [h_{1i}\ h_{2i}]^{transposed} = [s_0\ s_1]^{transposed}$, then ϖ enters into the *add* 4 from the middle pair of "and" gates, whereas from the other two pairs of the "and" gates 2 zeros enter into *add* 4. The *add* 4 then adds $\varpi, 0, 0, r_i$ and yields the ith coordinate of the transmitted codeword c.

4. Conclusion

In this study, we have established that the field of the lowest order over which there exists an 1-error correcting [5,3] code is a quaternary field. We also found that there exists only one distinct [5,3,3] quaternary code up to equivalence. This code is capable of 1-error correction in 64 distinct codewords. We devised a method to implement this quaternary 1-error correcting code in a binary data transmission system. We have simulated the designs of the encoder and the decoder using '*MODELSIM*' synthesized the logics using 'XILINX' and found the devices working correctly. We wish to analyze the efficiency and other characteristics of this code and related circuit designs in a different study.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed equally and significantly in writing this article. All the authors read and approved the final manuscript.

References

[1] V. Pless (2003), *Introduction to the Theory of Error Correcting Codes*, Wiley; Student Edition, John Wiley & Sons (Asia) Pte. Ltd., Singapore.

[2] R. Hill (1986), *A First Course in Coding Theory*, The Oxford University Press, Oxford, UK.

[3] W.C. Huffman and V. Pless (2003), *Fundamentals of Error Correcting Codes*, Cambridge University Press, New York.