A study on the VHDL Implementation of a RS coder for a FTS transceiver[1]

Wooshik Kim[1], Jun Seok Lim[2], Steve Yoon[3]
[1] Department of Information and Communication Engineering
[2] Department of Electronics Engineering
[3] Department of Aerospace Engineering
Sejong University
98 Kunja-dong, Kwangjin-ku. Seoul, Korea 143-747
wskim@sejong.ac.kr
Phone: +(822) 3408-3199
Mobile: (+8211) 9177-6342

Abstract: A FTS (Flight Termination System) is a system that resides in a flying object such as a rocket, unmanned airplane, helicopter, missile, etc., receives commands from ground stations or detects coordinates automatically, and accomplishes a destruction command in case the object does not follow the presumed orbit. In this paper, we address the implementation of a communication modem for the FTS modem. We present general theory, simulation results using Matlab, and several results on the implementation using VHDL.

Key word. FTS (Flight Termination System), VHDL, RS (Reed – Soloman) code

1. INTRODUCTION

A FTS (Flight Termination System) is a system that resides in a flying object such as a rocket, unmanned airplane, helicopter, missile, etc., receives commands from ground stations, and accomplishes destruction command in case the launched vehicle does not follow the presumed orbit. In some FTS, the flying object automatically calculates its orbit and may make a decision. For the FTS modem to function properly, the FTS should have a reliable communication of information over very hostile environments such as multipath environments and long range such as several thousands of kilometers. Thus the modem should have high-quality performances. In this paper, we address 2 issues:

1. Development of general concepts of FTS's and a prototype modem
2. Develop RS coder/decoder and their simulation using VHDL for implementing on a hardware platform such as FPGA, CPLD, etc.

This paper is organized as follows: In Section 2, we address the general block diagram of FTS modem. In Section3, we consider the implementation of RS encoder and decoder, including matlab simulation and VHDL simulation. In Section 4, we present simulation results. Finally, we present a summary in Section 5.

2. STRUCTURE OF A FTS MODEM

About the first issue, we present a general block diagram of a FTS modem in Figure 1.

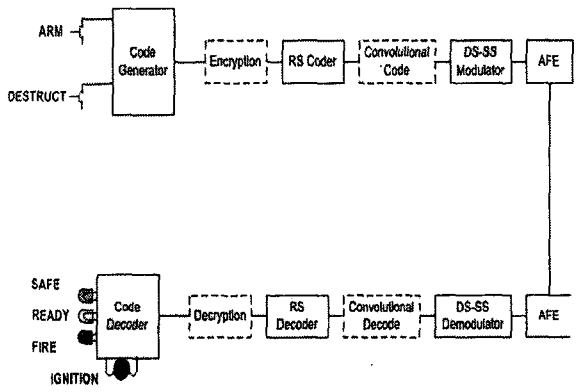


Figure 1 General Block Diagram of a FTS

As we can see in this figure, the modem composed of several blocks. In the transceiver, the first component is the code generator. This module is a state machine that gets the signal from the attached two buttons, ARM and DESTRUCT, and sends a coded command accordingly. In case the button has pushed unintentionally, or there are severe noises in the communication channel, the receiver does not decipher the command or decipher wrongly. To prevent this, the code generator is equipped with some special algorithm such as the 'Counter Model' [1] or State machine.

The next component is the encryption part. In the Special Report RS-38 on E-FTS, this function is implemented in the command using triple DES encryption method. The next component is RS encoder. This component now becomes a mandatory component in the modem, especially high speed modem such as ADSL, VDSL, IMT-2000, etc. In the next section, we consider this component. The next component may be a convolutional code. This component is not mandatory, however, this can improve the performance of the mode very much. The next component is modulation block. The

[1] This work is supported in part by the ABRC (Advanced Biometric Research Center) and the FTS project sponsored by KOSEF.

candidates of the modulation methods can be various analog modulation methods such as FM, AM, or digital modulation methods such as Frequency Shift Keying (FSK), CPFSK, or Enhanced High Alphabet (Hi Alpha). Later, we are going to use a Spread Spectrum method. The last component of the transmitter is the AFE, Analog Front End. This component transforms the digital data into analog signal modulated into the pre-assigned frequency.

3. RS ENCODER/DECODER

About the second issue, the FTS modem has several requirements. One of the most important is to have a reliable communication over hostile communication environments. To make the FTS meet the requirement, we need to use various coding algorithms for recovering lost commands and information that may occur during the transmission. Among these methods are CRC, Reed-Solomon Code, Convolutional code, and Turbo code, etc. Among these, as a first coding method, we consider RS (Reed-Solomon) coder.

As a prototype, we consider RS(31,21) code. This code has total 31 symbols, 155 bits of information. This case the symbol is composed of 5 bits and its elements are defined over GF(32). This code has 21 symbols, 105 bits of original information and 10 symbols, 50 bits of parity check symbols. This code can correct maximum 5 symbols, 25 bits.

3.1 RS Encoder

The main idea of the RS encoder is the attachment of a parity symbols after the information codes. The general equation of this is as follows

$$p(X) = i(X)X^{n-k} \bmod g(X)$$

This is basically the polynomial division. The polynomial division can be implemented using digital circuits. The block diagram may be [2]

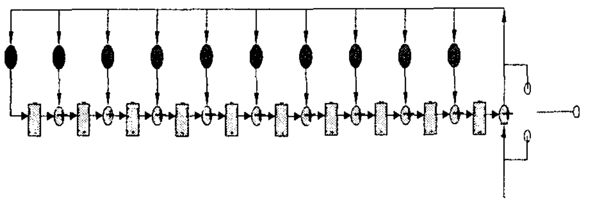


Figure 2 The Block Diagram of RS Encoder

3.2 RS Decoder

The next figure shows the general block diagram of RS decoder [2].

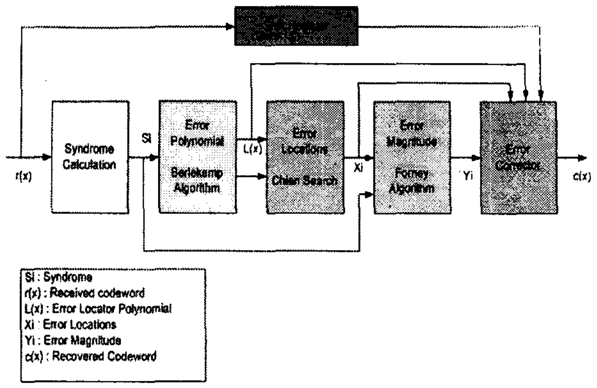


Figure 3 Block Diagram of RS Decoder

In general, the RS Decoder composed of a Syndrome calculation Module, a Finding Error Polynomial Module, a Finding Error Location Module, a Calculating Error Magnitude Module, and an Error Correction Component. In implementing this on Hardware device such as CPLD, FPGA, etc, we need another module, memory or registers that can hold the input code temporarily until they are combined with corrected errors if any. In this section, we consider each of the module.

A. Syndrome Computation

The algorithm of the syndrome computation is nothing but a finding the remainder. The equations for calculating syndrome are given as

$$S_1 = r(\alpha) = \sum_{i=0}^{M-1} r_i \alpha^i$$

$$S_2 = r(\alpha^2) = \sum_{i=0}^{M-1} r_i \alpha^{2i}$$

$$\vdots$$

$$S_v = r(\alpha^v) = \sum_{i=0}^{M-1} r_i \alpha^{vi}$$

If we implement this algorithm on a hardware, the block diagram may be [2]

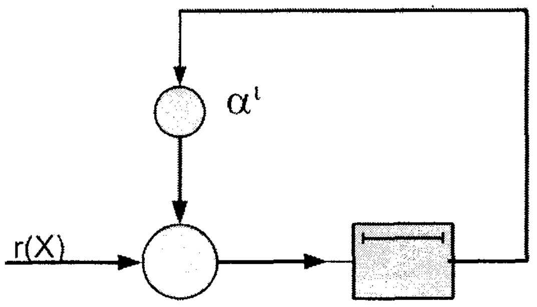


Figure 4 Block Diagram for Calculating Syndrome

B. Finding Error Polynomial

If we have found the syndrome, then we can get the error polynomial. The most popular algorithm for finding the error polynomial is the Berlekemp-Massey algorithm. We also consider this algorithm. The block diagram of the algorithm is give as [2]

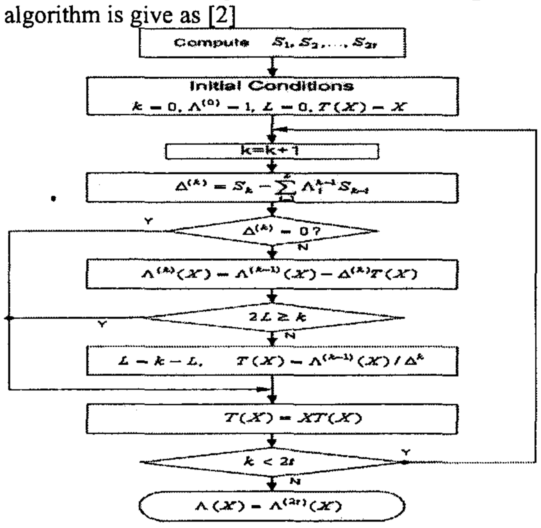

Compute $S_1, S_2, \ldots, S_{2t}$
Initial Conditions
$k = 0, \Lambda^{(0)} = 1, L = 0, T(X) = X$
$k = k+1$
$\Delta^{(k)} = S_k - \sum \Lambda_i^{(k-1)} S_{k-i}$
$\Delta^{(k)} = 0?$
Y
N
$\Lambda^{(k)}(X) = \Lambda^{(k-1)}(X) - \Delta^{(k)} T(X)$
$2L \geq k$
Y
N
$L = k - L, \quad T(X) = \Lambda^{(k-1)}(X) / \Delta^k$
$T(X) = XT(X)$
$k < 2t$
Y
N
$\Lambda(X) = \Lambda^{(2t)}(X)$

Figure 5 Block Diagram of Berlekemp Algorithm

C. Finding of the Error Locations

The main idea of the finding error location is to find the roots of the error polynomial

$$\Lambda(X) = 1 + \Lambda_1 X + \Lambda_2 X^2 + \cdots + \Lambda_t X^t$$

In this pape, we use Chien search algorithm. The block diagram of this algorithm is given in the next figure.

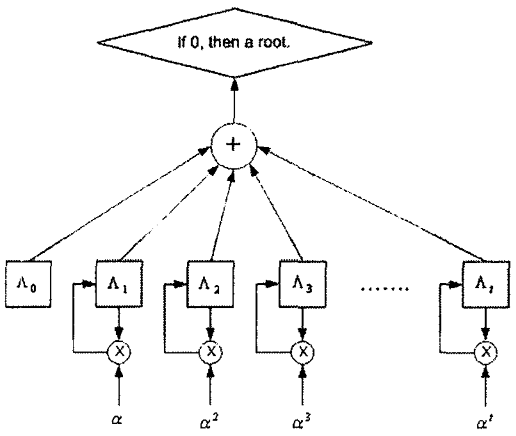


Figure 6 Block Diagram of Chien Search

D. Calculation of Error Magnitudes

The calculation of the error magnitude is very complex. The equation for the error magnitude is given as [2]

$$e_{ik} = \frac{-X_k \hat{\Omega}(X_k^{-1})}{\Lambda'(X_k^{-1})}$$

$$\hat{\Omega}(X) = Trunc\{\Omega(X)\} \; over \; X^{2t}$$

$$\Omega(X) = \Lambda(X)[1 + S(X)]$$

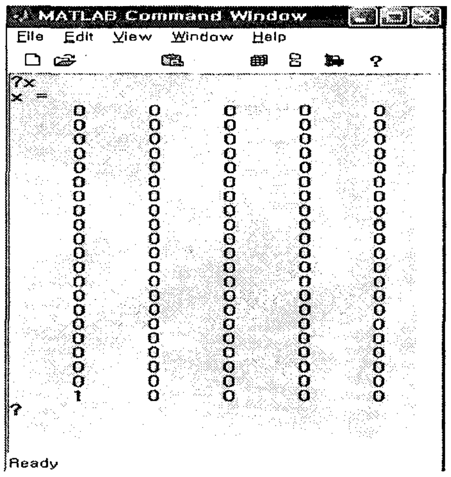
Using the results so far, In Section 4, we perform simulation and Implementation. The simulation is done using Matlab and the Implementation is done using VHDL.

4. SIMULATION

In this section, we consider both the Matlab and the VHDL simulation and compare these results. The VHDL simulation is done in Max Plus II, that is used for implementing on Altera devices.

4.1 RS Encoder

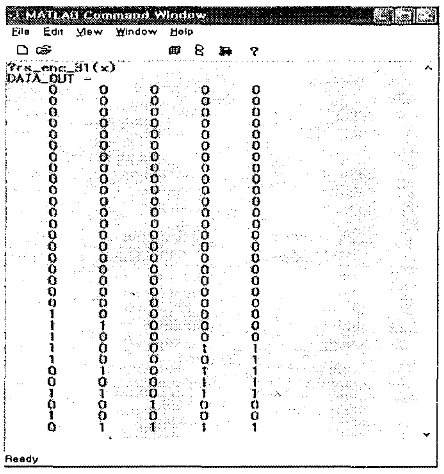
For RS encoder, we consider the information having 10 symbols. All the symbols have 5 bits of [0 0 0 0 0] except the last symbol. We assume that the last symbol has the value [1 0 0 0 0]. We show the result in the next figure.



```
?x
x =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     1     0     0     0     0
?
```

Figure 7 Information Symbols as an Input

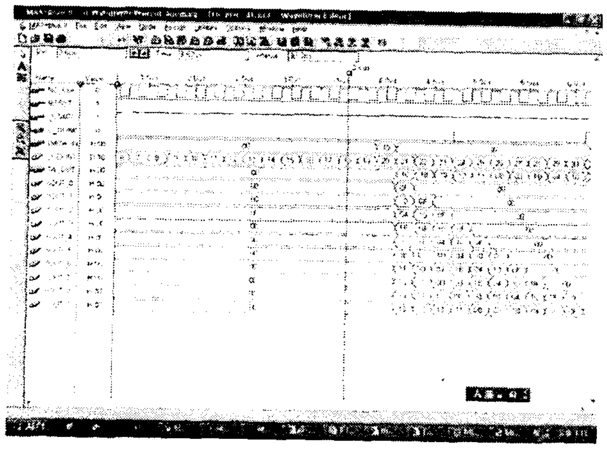
Since the last symbol is 1 while the others are all 0's, the coded output should be the coefficients of the parity check polynomial. In the next figure, we show that the simulation result using Matlab.



```
?rs_enc_31(x)
DATA_OUT =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     1     0     0     0     0
     1     1     0     0     0
     1     0     0     0     0
     1     0     0     1     1
     1     0     0     0     1
     0     1     0     1     1
     0     0     0     1     1
     1     1     0     1     1
     0     0     1     0     0
     1     0     0     0     0
     0     1     1     1     1
```

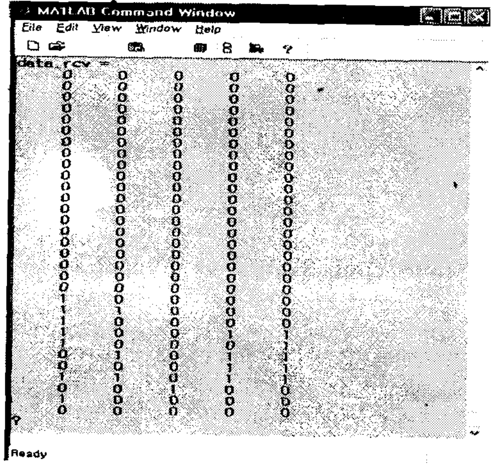
Figure 8 Coded Signal of the RS Encoder

As we can see in this figure, the calculated parity check symbols are the same as the coefficients of the polynomial.



5.2 Simulation of RS Decoder

To perform a simulation, i.e., to see whether the developed algorithm worked well or not, we introduced an error intentionally to mimic the bad channel. In this simulation, we consider only one error. In the next figure, we show the input.

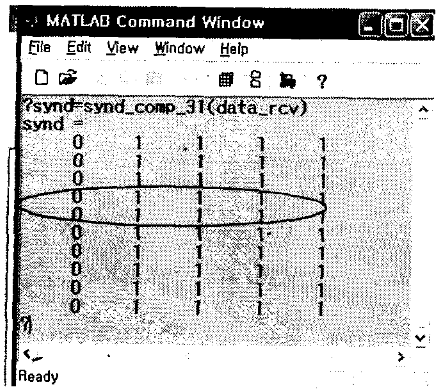


```
data_rcv =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     1     0     0     0     0
     1     1     0     0     0
     1     0     0     0     0
     1     0     0     1     1
     1     0     0     0     1
     0     1     0     1     1
     0     0     0     1     1
     1     1     0     1     1
     0     0     1     0     0
     1     0     0     0     0
     0     0     0     0     0
```

Figure 9 Received Code Assumed to Have an Error

A. Syndrome calculation

The first procedure is to calculate syndromes. If we use Matlab, then we can get the following results



```
?synd=synd_comp_31(data_rcv)
synd =
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
     0     1     1     1     1
```

Figure 10 Computed Syndrome Using Matlab

The next figure shows the simulation results using VHDL.

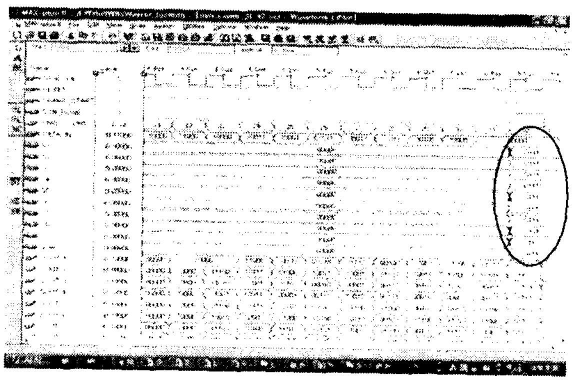
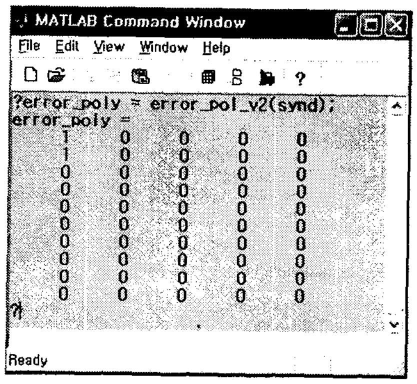


Figure 11 VHDL Simulation of Syndrome Computation

As we can see in the circled area, the computed syndromes are identical to the results in the Matlab.

B. Error Polynomial

Using these syndromes, we can calculate the error polynomial. The next figure shows the Matlab results and VHDL results.



```
?error_poly = error_pol_v2(synd);
error_poly =
     1     0     0     0     0
     1     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Figure 12 Computed Error Polynomial Using Matlab

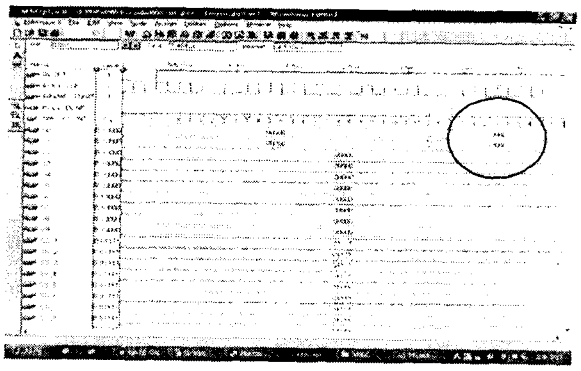


Figure 13 Simulated Error Polynomial using VHDL

In these two pictures, we can see that the error polynomial is obtained correctly.

B. Error Location

The error location can be implemented using VHDL. As we can expect from the assumption, the location of the error should be the last symbol, i.e., 31. The next figure shows the result.

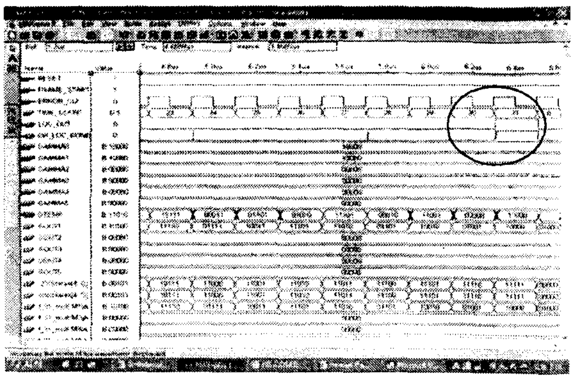
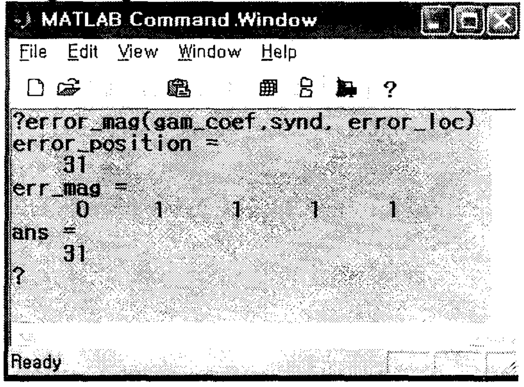


Figure 14 Result of Simulation of Finding Error Location

C. Finding Error Magnitude

The Calculation of the error magnitude can be done using the algorithm.



```
?error_mag(gam_coef,synd, error_loc)
error_position =
    31
err_mag =
    0    1    1    1    1
ans =
    31
?
```

Figure 15 Calculated Error Magnitude and Position in Matlab Simulation

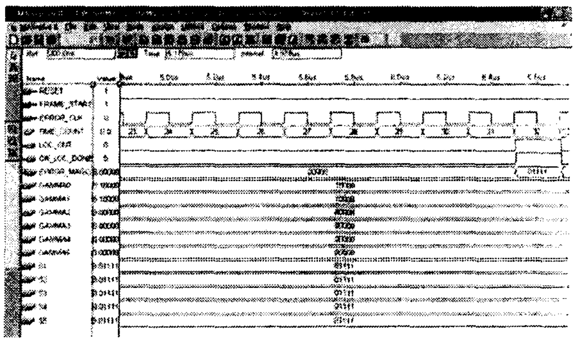


Figure 16 Simulation Result Using VHDL

D. Integration

Integrating these results on the designs by components, we get the complete result. In this figure, the circled area is the corrected final results. As we can see in this figure, the RS decoder has been implemented correctly.

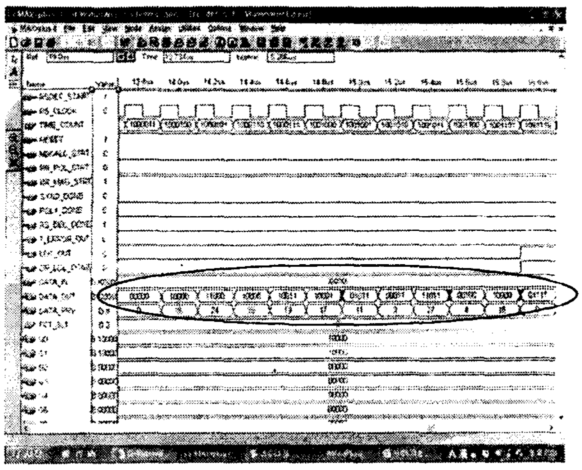


Figure 17 Summary of Results

5. SUMMARY

In summary, in this paper, we address the concept of FTS and its hardware implementation using VHDL. We developed a state machine model of a code generator and decoder to avoid accidental mishaps. Also, we presented a block diagram model of FTS modem. We have done simulation of RS coder/decoder using VHDL and developed a fast RS coder that can perform the correction.

In implementing a complex system on a hardware such as CPDL, FPGA, etc., the complexity and speed is the two most important factors. In terms of speed, the most crucial part is multipliers. In implementing this RS coder/decoder, we need multipliers over Galois Field. To minimize the time to compute, we need a fast multiplier and inverter. A next research may be the implementation of fast multiplier.

REFERENCES

[1] RCC, "Enhanced Flight Termination System Study Phase I -IV Reports", Special Report RS-38.

[2] S. Wicker, "Error Control Systems For Digital Communication And Storage", Prentice Hall.

[3] S. Lin, D. Costello, "Error Control Coding", Prentice Hall

[4] Kyung Ok Shin, "Design of Multiplier for Digital Signal Processing", IDEC Lecture Notes