

Implementación de un Codificador y Decodificador Reed Solomon (204,188) en una Tarjeta FPGA con el Algoritmo de Euclides Modificado

Implementation of a Reed Solomon encoder and Decoder (204.188) on an FPGA card with Euclid's modified algorithm

Cristian Oñate^{1,*}, y Pablo Lupera Morillo^{1,†}

¹Escuela Politécnica Nacional, Ecuador.

cristian697@gmail.com; pablo.lupera@epn.edu.ec

Fecha de recepción: 16 de agosto de 2018 — **Fecha de aceptación:** 30 de octubre de 2018

Cómo citar: Oñate, C., & Lupera, P. (2018). Implementación de un Codificador y Decodificador Reed Solomon (204,188) en una Tarjeta FPGA con el Algoritmo de Euclides Modificado. *Journal of Science and Research: Revista Ciencia e Investigación*, 3(JIEE2018), 35-42. <https://doi.org/10.26910/issn.2528-8083vol3issJIEE2018.2018pp35-42p>

Resumen—En este artículo se realiza un análisis de la teoría del código Reed-Solomon, para luego describir específicamente la codificación y decodificación del código Reed-Solomon (204, 188) en VHDL. Posterior a eso, se implementó el código descrito en la tarjeta FPGA Virtex 5 XUPV5-LX110T. El diseño del codificador se lo realizó usando su arquitectura genérica, mientras que para el decodificador se usaron como base los algoritmos de Euclides Modificado, Chien y Forney. Finalmente, se verificó su correcto funcionamiento y se determinó la cantidad de recursos utilizados de la tarjeta al implementar el algoritmo estudiado.

Palabras Clave—Algoritmo de Euclides Modificado, Corrección de errores, Reed Solomon.

Abstract—This article performs an analysis of the Reed-Solomon code theory, and then specifically describes the coding and decoding of the Reed-Solomon code (204, 188) in VHDL. After that, the code described in the Virtex 5 XUPV5-LX110T FPGA card was implemented. The encoder design was made using its generic architecture, while the decoder was used as the basis of the algorithms of Euclid Modified, Chien and Forney. Finally, the correct operation was verified and the amount of resources used on the card was determined when implementing the studied algorithm.

Keywords—Euclid's Modified Algorithm, Bug fixes, Reed Solomon.

INTRODUCCIÓN

EN el intercambio de información a través de un canal de comunicaciones poco confiable o ruidoso, es inevitable encontrarse con errores en la recepción, esto es inadmisibles si se pretende brindar una comunicación confiable y de calidad entre un emisor y un receptor, por lo que es necesario el uso de técnicas de codificación para el control de los errores (Castiñeira y Farrell, 2012).

En 1960 Irvin S. Reed y Gustave Solomon publicaron un artículo titulado "Polynomial Codes over Certain Finite Fields" presentando la creación del código que lleva sus nombres. Reed Solomon es un código usado para la detección y corrección de errores. La aplicación de este código fue complementada con la invención del algoritmo decodificador creado por Elwyn Berlekamp. Su primera aplicación significativa fue la codificación de imágenes digitales en la sonda espacial Voyager (Serrano, 2004).

*Estudiante del Departamento de Electrónica, Telecomunicaciones y Redes de Información.

†PhD. en Ciencias Técnicas.

En la actualidad tiene un amplio rango de aplicaciones en comunicaciones digitales y es usado para corregir errores en muchos sistemas de comunicación como: Wifi, Wimax (Lujan, Almagro, Cabrera, Suardíaz y Cerdan, n.d.), CDs y sondas espaciales (Bateman, 2003). También se destaca la aplicación del código Reed-Solomon en algunas tecnologías de comunicación digital, como: Digital Video Broadcasting (DVB), Integrated Services Digital Broadcasting Terrestrial (ISDB-T) y en Digital Audio Broadcasting (DAB) (Serrano, 2004).

MARCO TEÓRICO

Los códigos Reed Solomon se representan como RS(n,k), en donde los parámetros definidos se expresan de la siguiente manera (Serrano, 2004):

n : número de símbolos a la salida del codificador (longitud de la palabra código).

K : número de símbolos de información que van a ser codificados.

$2t = n - k$: número de símbolos de paridad añadidos dentro de los n símbolos.

$t = (n - k) / 2$: número máximo de errores que puede corregir el código.

Normalmente se los representa de manera sistemática, eso quiere decir que después de la codificación la palabra original se la puede distinguir claramente en la palabra codificada (Serrano, 2004) (ver Figura 1).

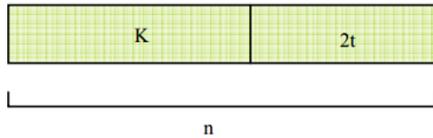


Figura 1: Parámetros del código RS(n,k)

Fuente: (Serrano, 2004).

Codificación Reed Salomon

La codificación tiene como función principal generar la palabra código, para esto la codificación sistemática añade n-k símbolos de paridad a la palabra original que ingresa al codificador, así ingresan k símbolos de información y salen n símbolos y esta es la palabra código. Los símbolos de paridad se los calcula multiplicando la palabra original por el polinomio X^{n-k} y este resultado se lo divide para el polinomio generador $g(x)$, el residuo de esta división es el conjunto de los símbolos de paridad que se añade a la palabra original (Castiñeira y Farrell, 2012).

El polinomio generador es un polinomio de grado X^{n-k} sobre el campo de Galois, cuyo valor dependerá del campo sobre el que se esté trabajando. El polinomio generador $g(x)$ de manera general se lo calcula de la siguiente manera (Castiñeira y Farrell, 2012).

$$g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)...(x + \alpha^{2t}) \quad (1)$$

Se presenta la descripción matemática de la codificación sistemática.

$$c(x) = m(x).X^{n-k} + [m(x).X^{n-k}] \text{mod} g(x) \quad (2)$$

$m(x)$: polinomio de la palabra original recibida a la entrada del codificador,

$c(x)$: la palabra codificada (palabra código),

$g(x)$: es el polinomio generador,

X^{n-k} : es un término de grado $n - k$

Decodificación Reed Salomon

El proceso de decodificación consiste en tomar la palabra enviada por el codificador y recuperar la palabra original. Si la palabra que se recibe tiene errores, el código Reed Solomon se encargará de detectar y corregir los errores que se produjeron en la transmisión, como este proceso es el inverso de la codificación ingresarán n símbolos al decodificador y se obtendrán los k símbolos correspondientes al mensaje original que se envió (Castiñeira y Farrell, 2012). El proceso de decodificación se lo describe a continuación:

Cálculo del síndrome

La manera más sencilla de calcular el síndrome es evaluar la palabra codificada $c(x)$ en cada una de las raíces del polinomio generador $g(x)$, se necesitan $2t$ síndromes para poder detectar y corregir t errores, con esto se calculará el polinomio síndrome $S(x)$, que permitirá determinar si la palabra código recibida tienen errores o no (Castiñeira y Farrell, 2012). El cálculo del polinomio síndrome se lo puede expresar matemáticamente de la siguiente manera:

$$S_{i-1} = c(\alpha^i), i = 1, 2, 3 \quad (3)$$

$$S(x) = S_0 + S_1X + S_2X^2 + \dots + S_{2t-1}X^{2t-1} \quad (4)$$

Si el polinomio síndrome es distinto de cero, esto indica que existen errores en la palabra recibida por lo que se deberá iniciar el proceso de detección y corrección de errores, detallado a continuación:

Detección y corrección de errores

Para la detección de errores es necesario calcular el polinomio localizador de errores y el polinomio magnitud de errores, para esto se pueden usar varios algoritmos como el de Berlekamp, Berlekamp-Masey, Euclides extendido o Euclides modificado, debido a las ventajas que ofrece el algoritmo de Euclides modificado respecto a la facilidad de ejecución y uso de recursos, en este proyecto se usa el algoritmo de Euclides modificado, el cual se lo ejecuta de la siguiente manera (Serrano, 2004):

Algoritmo de Euclides modificado

Este algoritmo está basado en el algoritmo de Euclides, el cual es usado para calcular el máximo común divisor de dos factores, sin embargo el algoritmo de Euclides modificado evita el cálculo de los cocientes en cada división, por lo que su ejecución es más rápida y consume menos recursos. A continuación, se describen las ecuaciones y sus condiciones iniciales para el uso del Algoritmo de Euclides modificado (Serrano, 2004):

$$\mu_0(x) = 1$$

$$\lambda_0(x) = 0$$

$$R_0(x) = X^{n-k}, \text{ término de grado } n - k$$

El polinomio magnitud de errores se calcula así (Serrano, 2004):

$$R_i(x) = [\sigma_{i-1}b_{i-1}R_{i-1}(x) + \overline{\sigma_{i-1}}a_{i-1}Q_{i-1}(x)] - X^{l(i-1)}[\sigma_{i-1}a_{i-1}Q_{i-1}(x) + \overline{\sigma_{i-1}}b_{i-1}R_{i-1}(x)] \quad (5)$$

El polinomio localizador de errores se determina de la siguiente manera (Serrano, 2004):

$$\lambda_i(x) = [\sigma_{i-1}b_{i-1}\lambda_{i-1}(x) + \overline{\sigma_{i-1}}a_{i-1}\mu_{i-1}(x)] - X^{l(i-1)}[\sigma_{i-1}a_{i-1}\mu_{i-1}(x) + \overline{\sigma_{i-1}}b_{i-1}\lambda_{i-1}(x)] \quad (6)$$

Los términos $Q_i(x)$ y $\mu_i(x)$ son expresiones complementarias que ayudan al cálculo de $\lambda_i(x)$ y $R_i(x)$

$$Q_i(x) = \sigma_{i-1}Q_{i-1}(x) + \overline{\sigma_{i-1}}R_{i-1}(x) \quad (7)$$

$$\mu_i(x) = \sigma_{i-1}\mu_{i-1}(x) + \overline{\sigma_{i-1}}\lambda_{i-1}(x) \quad (8)$$

Donde a_{i-1} y b_{i-1} son los coeficientes de mayor grado de los polinomios $R_{i-1}(x)$ y $Q_{i-1}(x)$ respectivamente.

$$l_{i-1} = \text{grado}(R_{i-1}(x)) - \text{grado}(Q_{i-1}(x));$$

$$\sigma_{i-1} = 1 \quad \text{si } l_{i-1} \geq 0,$$

$$\sigma_{i-1} = 0 \quad \text{si } l_{i-1} < 0.$$

La iteración del algoritmo termina cuando se cumple que $(\lambda_i(x)) > \text{grado}(R_i(x))$ o bien $\text{grado}(\mu_i(x)) > \text{grado}(Q_i(x))$, en ese momento el polinomio localizador $\sigma(x)$ será $\lambda_i(x)$ y el polinomio magnitud de error $\omega(x)$ será $R_i(x)$ (Serrano, 2004).

Una vez concluido el algoritmo de Euclides Modificado, se procede con la ejecución del algoritmo de Chien cuya función principal es encontrar la posición del error en la palabra código recibida. Para esto, se calculan las raíces del polinomio localizador de errores $\sigma(x)$. La manera más fácil de calcular sus raíces es reemplazando la variable x por todos los elementos del campo finito de Galois, excepto el 0. Si α^i es una raíz del polinomio $\sigma(x)$, la posición del error será $(2^m - 1) - i$, 2^m para este caso es 256, este valor dependerá del campo de Galois que se esté usando para el código (Serrano, 2004).

El último algoritmo a ejecutar es el algoritmo de Forney, este permite calcular el valor del error localizado mediante la evaluación de la siguiente fórmula (Serrano, 2004):

$$e = \frac{\omega(x)}{\sigma'(x)} \quad (9)$$

Corrección de errores

Una vez ejecutados los algoritmos de Euclides modificado, Chien y Forney se ha encontrado el polinomio error e(x).

$$e(x) = (\text{valor hallado con Forney})X^{\text{Posicion hallada en Chien}} \quad (10)$$

Si existe más de un error, cada uno tendrá su valor y posición correspondiente. La corrección de errores se lo hace realizando la operación XOR entre el polinomio error y la palabra recibida (Castiñeira y Farrell, 2012).

$$m(x) = c(x) + e(x) \quad (11)$$

Este método de codificación es general y puede ser aplicado para códigos Reed Solomon de cualquier magnitud.

Descripción en VHDL

Para la descripción del código Reed Solomon (204,188) se usará el lenguaje VHDL, que es un acrónimo de VHSIC Hardware Description Language, en donde VHSIC es otro acrónimo de Very High Speed Integrated Circuit (Chu, 2006). Este lenguaje es ampliamente usado en la industria y permite realizar una descripción de circuitos síncronos y asíncronos (Sanchez, n.d.).

La descripción en VHDL inicia con el diseño de un UART (Universal Asynchronous Receiver-Transmitter), el cual usará el estándar RS-232 para la comunicación entre el computador y la FPGA (Chu, 2006).

Para la descripción del codificador Reed Solomon se usará una arquitectura genérica que puede ser usada en cualquier codificador Reed Solomon sin importar los valores de n y k (Sandoval y Fedon, 2008). Esta arquitectura se muestra en la Figura 2.

A continuación, se definen los elementos más importantes en la arquitectura genérica y su funcionamiento.

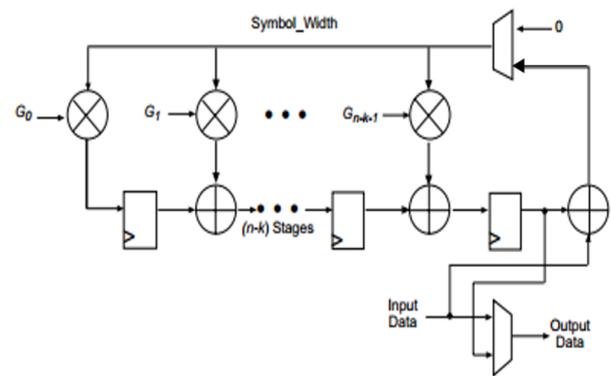


Figura 2: Arquitectura genérica de un codificador Reed Solomon (n,k)

- Multiplicación en el campo de Galois.
- Suma en el campo de Galois.
- Registro de memoria de 8 bits (Flip flop tipo D)

G_n : Coeficientes del polinomio generador $g(x)$
 El circuito inicia su proceso con el ingreso del primer símbolo de la palabra original, el cual se sumará con el valor almacenado en el registro $n - k - 1$ que será el valor 0, ya que es el valor inicial en cada registro, este resultado es multiplicado por cada coeficiente del polinomio generador $g(x)$, $G_0, G_1, \dots, G_{n-k-1}$. El resultado entre el primer elemento y el coeficiente G_0 es almacenado en su correspondiente registro de memoria, mientras que los demás resultados de cada multiplicación en cambio son sumados con el dato almacenado previamente en cada registro, que como se sabe tiene el valor inicial 0, realizada la suma el resultado se almacenará en su registro correspondiente.

Luego ingresará el segundo símbolo y se ejecutará el mismo proceso, el proceso se repetirá hasta que todos los símbolos de la palabra original hayan ingresado, una vez que se han procesado todos los símbolos, los últimos valores almacenados en cada registro se agregarán a la palabra original para obtener la palabra completa codificada.

Una vez descrito el funcionamiento de cada señal, en la Figura 3 se muestra la arquitectura del codificador Reed Solomon (204,188) con UART.



Figura 3: Esquemático del codificador con UART instanciado ISE

El diseño del decodificador presenta una alta complejidad, por lo que se ha dividido su descripción en 3 bloques, el primero encargado de calcular el polinomio síndrome, el segundo ejecutará el algoritmo de Euclides modificado para

hallar los polinomios localizador y magnitud de error, y el último bloque, será el encargado de hallar la magnitud y posición de los errores y realizar la corrección de los mismos, esta arquitectura se la muestra en la Figura 4. Para el cálculo

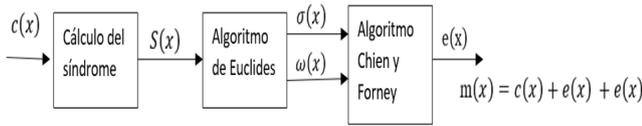


Figura 4: Arquitectura para el decodificador Reed Solomon (n, k)

del síndrome se usa la arquitectura que se muestra en la Figura 5, que se encarga de evaluar la palabra recibida en el elemento α^i para hallar el coeficiente S_{i-1} del polinomio síndrome. Ya que el polinomio síndrome consta de 16 coeficientes, en VHDL se implementan 16 circuitos iguales cada uno con su respectivo α^i .

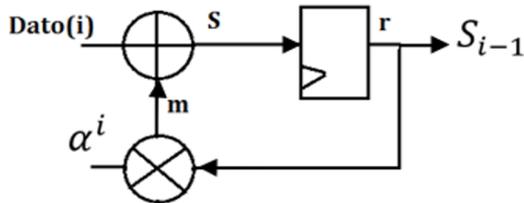


Figura 5: Arquitectura para el cálculo del síndrome en ISE

El bloque del algoritmo de Euclides modificado está implementado mediante la máquina de estados que se muestra en la Figura 6. En la máquina de estados se ejecutan las iteraciones necesarias para hallar los polinomios localizador y magnitud de errores.

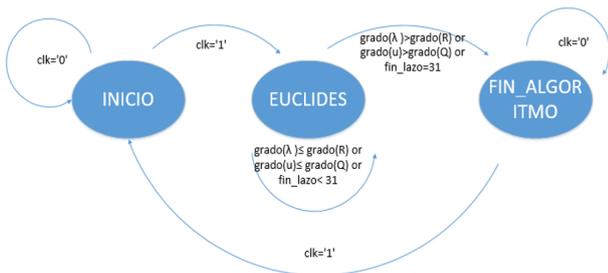


Figura 6: Máquina de estados para la ejecución del algoritmo de Euclides modificado.

Como se explicó anteriormente el algoritmo de Chien detecta la posición del error y esto se logra evaluando el polinomio localizador en un α^i , por lo que su arquitectura se ajusta a lo mostrado en la Figura 7 y será muy similar a la usada para el cálculo del polinomio síndrome. Esta arquitectura se repetirá 3 veces en la descripción para evaluar el polinomio localizador de errores, su derivada y el polinomio magnitud de errores. Una vez evaluados los 3 polinomios, se toma el valor calculado

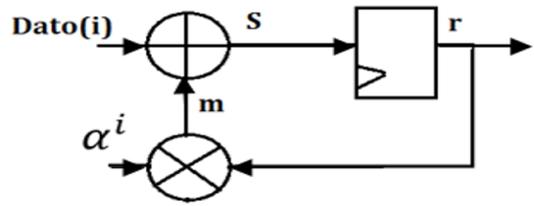


Figura 7: Arquitectura usada para el algoritmo de CHIEN.

después de evaluar el polinomio localizador de errores y si este es cero, se aplica el algoritmo de Forney para calcular la magnitud del error, si no es así entonces no existirá error en esa posición y no será necesario ejecutar el algoritmo, para este caso se asigna el valor 00000000 en esa posición. Este proceso se lo observa en la Figura 8, la cual describe la arquitectura usada para el algoritmo de Forney. Realizada la descripción

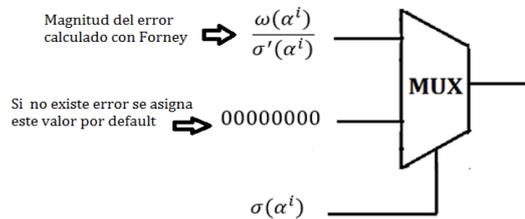


Figura 8: Arquitectura usada para el algoritmo de Forney.

en VHDL del decodificador, se instancia con su respectivo transmisor y receptor serial para obtener el decodificador Reed Solomon (204,188) con UART mostrado en la siguiente Figura 9.

Una descripción detallada de la implementación de los algoritmos de codificación y decodificación que se desarrollaron, se puede encontrar en la referencia [8] que pertenece a los mismos autores del presente artículo.

RESULTADOS Y DISCUSIÓN

En esta sección se muestran los resultados obtenidos luego de la implementación en la FPGA del código descrito en VHDL, tanto para el codificador como para el decodificador Reed Solomon (204, 188).

Evaluación y pruebas de la implementación del codificador Reed Solomon (204, 188) con UART

En la siguiente la Tabla 1 se muestran los recursos utilizados para la implementación del codificador Reed Solomon (204,188) en la tarjeta FPGA empleada.

A continuación, se muestran los resultados obtenidos en la interfaz gráfica tras ejecutar las pruebas prácticas del codificador Reed Solomon (204,188) en la FPGA, en este ejemplo se enviaron 188 símbolos con valor 01_H, y se obtiene el dato codificado mostrado en la Figura 10. En las Figuras 11 y 12 se puede ver el dato codificado obtenido en Matlab e ISE respectivamente luego de ingresar 188 símbolos con valor 01_H.

