Control de Errores en una Estación Base de Comunicación Móvil Basado en Codificación de Bloques Lineales y Codificación Convolucional

Jaime Humberto Pech Carmona[1] y Karla Vázquez Ascención[2]

Resumen

Este trabajo consiste en la implementación de un sistema robusto de corrección y detección de errores para una estación base con tres mensajes de un tamaño limitado. Se considera en el documento una introducción a las características generales de los tipos de codificador mencionados, el criterio matemático que fundamenta la aplicación y los elementos requeridos en la implementación empleando un Arreglo de Compuertas Programables de Campo (FPGA). La implementación considera el manejo del procesador NIOS II y el uso del sistema operativo de tiempo real μc-OS. En la última sección se

Acerca de los autores...

[1]División de Ingeniería Electrónica, del Tecnológico de Estudios Superiores de Ecatepec.

[2]División de Ingeniería en Sistemas Computacionales del Tecnológico de Estudios Superiores de Ecatepec

contemplan los recursos matemáticos para la obtención de tasas de error de bit (BER) para los codificadores respectivos.

Abstract

This work consists of the implementation of a robust system of correction and error detection for a base station with three messages of a limited size. It is considered in the document an introduction to the general characteristics of the types of encoder mentioned, the mathematical criterion that bases the application and the elements required in the implementation using an arrangement of programmable floodgates Field (FPGA). The implementation considers the handling of the NIOS II processor and the use of the μc-C-OS real-time operating system. The last section includes mathematical resources for obtaining bit error rates (BER) for the respective encoders.

Palabras Clave.- Codificador de bloques, codificador convolucional, detección y corrección de errores.

***Index Terms**.- Block encoder, convolutional encoder, error detection and correction.*

Introducción

El objetivo de este proyecto es crear un programa mediante μc-OS que emule una estación base de un sistema de comunicaciones inalámbrico, tal como podría ser en una red celular. Dicho proyecto está constituido por dos procesos principales: un codificador convolucional ½ y un codificador de bloques (8,5), ambos componentes existen dentro de una estación base real de telecomunicación. El diagrama de bloques es el siguiente:

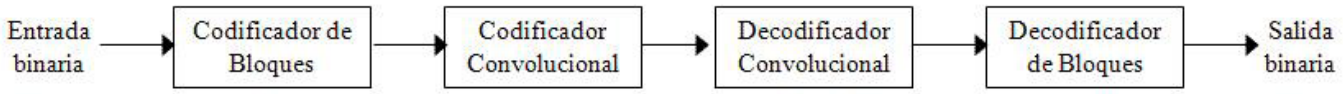


Figura 1

Diagrama de Bloques del Sistema emulado.

La finalidad de toda transmisión de información o datos reside en que el receptor reciba exactamente lo que se le envía desde un emisor dado. El principal problema se encuentra en que el medio de transmisión (canal) no presenta condiciones ideales, es decir, presenta condiciones adversas, tales como atenuación, distorsión, ruido, desvanecimiento, entre otros. De las características anteriores, es posible clasificar a los canales de acuerdo a la perturbación principal a la cual se somete la información, por lo que considerando un ruido aleatorio es posible valorar al canal como AWGN (Additive White Gaussian Noise).

En consecuencia a las condiciones ruidosas del medio, se requiere de un proceso mediante el cual sea posible identificar con la mayor exactitud al mensaje que ha sido enviado. A falta o en combinación con un ajuste dinámico de potencia, la propuesta radica en un proceso de codificación. Un codificador es un dispositivo hardware/software a través del cual la señal de información es modificada, normalmente agregando redundancia, de tal manera que sea posible con una mayor probabilidad recibir el mensaje tal y como fue generado por la fuente. Los tipos de codificadores anteriormente descritos son denominados codificadores de canal y existen dos tipos principales: codificador de bloques y codificador convolucional.

Para este proyecto, se utilizó como estación base el FPGA y el procesador Embebido NIOS II. Los elementos periféricos usados fueron:

- Switches SW14 al SW0 para introducir los mensajes a enviar.
- Leds para mostrar el resultado final.
- LCD para mostrar mensaje original y decodificado.
- Botones para que comience el programa y/o agregar errores

Una de las características de una estación base es que recibe varios mensajes o tareas al mismo tiempo para que sean procesados y de nuevo enviados por un canal inalámbrico. Esta característica es tomada para este proyecto mediante el uso de switches, en donde se detectarán hasta tres mensajes al mismo tiempo y serán procesados por el sistema que se muestra en la Figura 1. El presente documento se encuentra organizado de la siguiente manera: en la Sección 2 se explica la teoría de codificación de bloques y convolucional; en la Sección 3 el diseño de cada uno de los codificadores, que resultarán útiles en la Sección 4, donde se explica la implementación en el FPGA. Finalmente, se hará un análisis del sistema en cuanto a la probabilidad de error en bit.

I. Teoría de la Codificación

Teoría de Codificación de Bloques

Para realizar la codificación, es necesario establecer una matriz generadora *G*, la cual está compuesta por dos submatrices: una de paridad *P* y otra de identidad, cuya forma es como se muestra en la ecuación (1).

$$G = \left[P | I_k \right]$$

Donde:
G es la matriz generadora.
P es la matriz de paridad.
I es la matriz de identidad de *k* x *k* bits.

Así, el mensaje se forma por la ecuación (2).

$$\mathbf{U = mG}$$

Donde:
U es el mensaje codificado.
m es el mensaje a codificar de *k* bits.
G es la matriz generadora.

La palabra codificada *U* será proporcionada al modulador, para posteriormente ser enviada por el canal inalámbrico. En el receptor, una vez demodulada, la palabra código *U* es recibida, pero dado que es posible que contenga errores, es necesario contar con un mecanismo para detectar y corregir, en la medida de lo posible, la mayor cantidad de información original. A la palabra código recibida se le denominará *r* en adelante.

Una forma de saber si la palabra código *r* recibida contiene errores, es mediante el cálculo del síndrome, el cual se lleva a cabo conforme a la expresión (3).

$$S = rÇ^T = eH^T$$

Donde:
S es el síndrome.
r es la palabra código recibida.
H es la matriz de revisión de paridad.
e es el patrón de errores.

La matriz de revisión de paridad *H* se forma de manera similar que la matriz *G*. En la relación mostrada en la ecuación (4) se observa la conformación estructural de *H*.

$$H = \left[I_{n-k} \middle| P^T \right]$$

Donde:
H es la matriz de revisión de paridad.
P es la matriz de paridad.
I es la matriz de identidad de *n-k* x *n-k* bits.

Si el síndrome es igual a cero, significa que no hubo errores, de lo contario, es necesario detectar y corregir el mayor número posible de éstos. Para ello, se hace uso del patrón de errores *e* mostrado en la expresión (3). Así, *r* es corregida de la forma mostrada en la expresión (5):

$$\hat{U} = r + \hat{e}$$

Donde:
Û es la palabra código corregida.
R es la palabra código recibida.
ê es el patrón de error correspondiente al síndrome.

Finalmente, a partir de *Û* se puede recuperar el mensaje enviado, el cual corresponde a los últimos *k* bits.

Para cada código de bloques es posible estimar el número de errores que pueden ser detectados y corregidos. Para lo anterior, es necesario hacer uso de las ecuaciones (6), (7) y (8) que se muestran a continuación.

$$\varepsilon = d_{\min} - 1$$

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

$$d_{\min} \leq \frac{n\left(2^{k-1}\right)}{2^k - 1}$$

Donde:
ε es la capacidad de detección de errores de código de bloques.
t es la capacidad de corregir errores.
d_{min} es la distancia mínima del código.

Teoría de Codificación Convolucional

En la codificación de bloques, el codificador acepta un bloque de mensaje de *k* bits y genera una palabra de código de *n* bits. De este modo, las palabras de código se producen en un esquema de bloque por bloque. Por lo anterior, es necesario tomar provisiones en el codificador para retener en un buffer un bloque de mensajes completo antes de generar la palabra de código asociada, sin embargo para las ocasiones en las cuales los bits de información se encuentran en su modalidad serial, el buffer puede ser indeseable. Para las situaciones descritas, el empleo de la codificación convolucional es mejor candidato para el proceso de detección y corrección de errores. Un codificador convolucional genera bits redundantes utilizando convoluciones módulo 2, de allí el nombre que adopta.

Un codificador convolucional queda especificado por una trupla (*n*, *k*, *m*) donde:

- n es el número de bits de la palabra codificada.
- k es el número de bits de la palabra de datos.
- m es la memoria del código o longitud restringida (tamaño del registro de corrimiento).

La tasa del codificador está constituida por el cociente k/n.
El codificador de un código convolucional binario con tasa 1/n, medida en bits por símbolo puede considerarse como una máquina de estado finito que consiste en un registro de corrimiento de *M* etapas con conexiones preestablecidas a n sumadores módulo 2 (generadores) y a un multiplexor que pone en serie las salidas de los sumadores.

Una secuencia de mensajes de *L* bits produce una secuencia de salida codificada de longitud n(L+M) bits.

La tasa de código está dada por la ecuación (6).

$$r = \frac{L}{n(L+M)} \quad bits/símbolo$$

Comúnmente se tiene L >> M, por lo que la tasa de código puede simplificarse en la expresión (7).

$$r \approx \frac{1}{n} \quad bits/símbolo$$

La longitud de restricción de un código convolucional, expresada en términos de bits de mensaje, se define como el número de corrimientos sobre el cual un bit de un solo mensaje puede influir en la salida del codificador. En un codificador con un registro de corrimiento de *M* etapas, la memoria del codificador es igual a *M* bits de mensaje y se requiere K=M+1 corrimientos para que un bit de mensaje entre al registro de corrimiento y finalmente salga, por lo tanto la longitud de restricción del codificador es *K*.

Es posible generar un código convolucional binario con tasa k/n utilizando *k* registros de corrimiento independientes con conexiones preestablecidas con *n* sumadores módulo 2, un multiplexor de entrada y un multiplexor de salida. Los códigos convolucionales generados por el codificador descrito son códigos no sistemáticos.

Cada trayectoria que conecta la salida con la entrada de un codificador convolucional puede caracterizarse en términos de su respuesta al impulso, definida como la respuesta a esa trayectoria a un símbolo *1* aplicado a su entrada, con cada flip-flop en el conjunto codificador inicialmente en el estado cero.
De manera similar, es posible caracterizar cada trayectoria en términos de un polinomio generador, definido como la transformada de retorno unitario de la respuesta al impulso. De manera correspondiente, el polinomio generador de la i-ésima trayectoria está definido como se observa en la expresión (8):

$$g^{(i)}(D) = g_0^{(i)} + g_1^{(i)}D + g_2^{(i)}D^2 + \cdots + g_M^{(i)}D^M$$

Donde *D* denota la variable de retardo unitario.

El codificador convolucional completo se describe mediante un conjunto de polinomios generadores $\{g^{(1)}(D), g^{(2)}(D), \ldots, g^{(n)}(D)\}$.

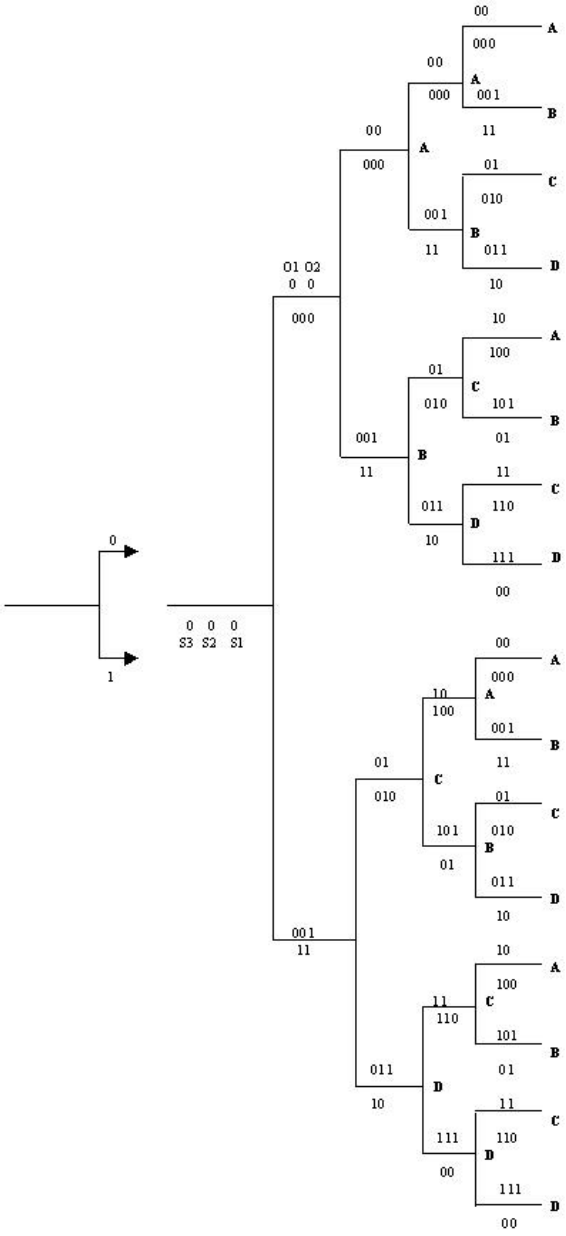


Figura 2

Diagrama de Árbol.

Tradicionalmente, las propiedades estructurales de un codificador convolucional se representan en forma gráfica utilizando uno de tres diagramas equivalentes:

- Árbol de código.
- Trellis (enramado).
- Diagrama de estados.

Diagrama árbol o árbol del código

Se refiere a la representación mediante un árbol binario con los distintos pesos y nodos del sistema (Figura 2). La profundidad del árbol es 2·(m-1), y el número de estados es $2^{(m-1)\cdot k}$. La interpretación del árbol del código es la siguiente:

- Hay dos ramas en cada nodo.
- La rama superior corresponde a una entrada de un *0*.
- La rama inferior corresponde a la entrada de un *1*.
- En la parte exterior de cada rama se muestra el valor de salida.
- El número de ramas se va multiplicando por dos con cada nueva entrada.

A partir del segundo nivel, el árbol se vuelve repetitivo. En realidad y considerando la justificación anterior, solo hay cuatro tipos de nodos: A, B, C y D. Estos tipos de nodos en realidad son estados del codificador. A partir de estos nodos, se producen los mismos bits de salida y el mismo estado.

Diagrama de estados

Emplea una máquina de estados de Mealy o Moore para plantear el comportamiento del sistema. A través de la máquina de estados se requiere desarrollar una tabla de verdad y realizar la codificación respectiva (Figura 3).

Figura 3

Diagrama de Estados.

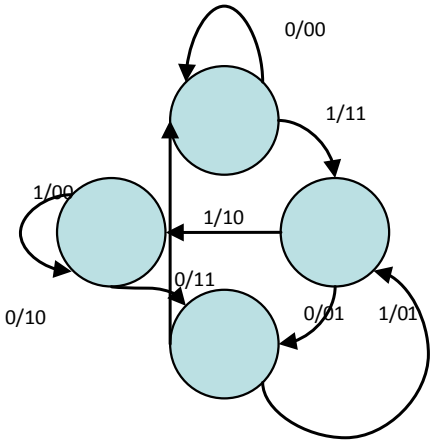


Diagrama de Trellis o enrejado

Es la forma más utilizada, porque permite realizar la decodificación de la manara más sencilla. El diagrama de Trellis es un diagrama en forma de red, donde cada línea horizontal corresponde a uno de los estados del codificador. Cada línea vertical se correspondería con uno de los niveles del árbol del código, es decir con un código convolucionado.

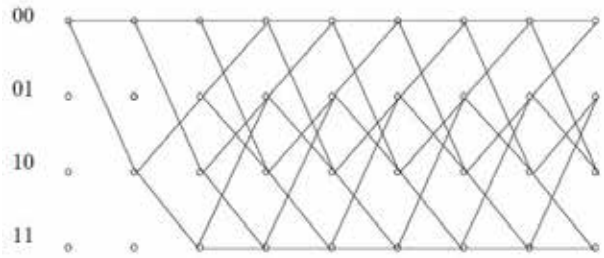


Figura 4

Diagrama de Trellis.

Se parte del estado inicial del codificador en el primer nivel del árbol. A partir de aquí se trazan dos líneas desde este estado. Una para el caso de que la siguiente entrada fuera un 0 (superior) y otra para el caso de que fuera un 1 (línea inferior). Estas líneas irán hasta el siguiente nivel del árbol al estado en el que queda el codificador después de haber codificado las correspondientes entradas. Encima de cada una de estas líneas se escribe la salida del codificador para esa codificación (Figura 4).

II. Teoría de Codificadores para la Estación Base

Cálculos para el Codificador de Bloques (8,5)

Como ya se había mencionado en la introducción, el tipo de codificación implementado es un codificador de bloques (8,5), donde n = 8, k = 5, es decir, cada mensaje está conformado por 5 bits, el cual será codificado en una palabra código de 8 bits. Para el codificador, es necesario conocer la matriz generadora G para que a partir de ella y de los posibles mensajes pueda ser obtenida la ecuación para obtener cada elemento de la palabra codificada. La matriz generadora de la aplicación se aprecia en la expresión (9).

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

De esta manera, para formar la palabra código para el objetivo inicial, se tiene lo siguiente:

$$\begin{bmatrix} x_4 & x_3 & x_2 & x_1 & x_0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} u_7 & u_6 & u_5 & u_4 & u_3 & u_2 & u_1 & u_0 \end{bmatrix}$$

Al analizar la operación (10), se pueden obtener las funciones correspondientes a cada uno de los elementos del mensaje codificado, las cuales se muestran a continuación en la expresión (11). Estas funciones son usadas en el programa que se mostrará en la Sección III.

$$
\begin{aligned}
u_7 &= x_3 \\
u_6 &= x_4 \oplus x_0 \\
u_5 &= x_1 \oplus x_0 \\
u_4 &= x_4 \\
u_3 &= x_3 \\
u_2 &= x_2 \\
u_1 &= x_1 \\
u_0 &= x_0
\end{aligned}
\tag{11}
$$

Del lado del decodificador, es necesario conocer la matriz *H* para obtener las funciones correspondientes al síndrome *S* y el patrón de errores *e* indicado en las expresiones (3) y (5). En las expresiones (12) a (15) se muestran las matrices y las fórmulas correspondientes para la implementación de este decodificador.

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

$$
\begin{aligned}
&\begin{bmatrix} s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 \end{bmatrix} \\
&= \begin{bmatrix} u_7 & u_6 & u_5 & u_4 & u_3 & u_2 & u_1 & u_0 \end{bmatrix} \cdot \\
&\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
s_2 &= r_7 \oplus r_3 \\
s_1 &= r_6 \oplus r_4 \oplus r_0 \\
s_0 &= r_5 \oplus r_1 \oplus r_0
\end{aligned}
$$

$$
\mathbf{e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
$$

$$
\begin{aligned}
e_7 &= s_2 s_1 s_0 \\
e_6 &= s_2 s_1 \overline{s}_0 \\
e_5 &= s_2 \overline{s}_1 s_0 \\
e_4 &= s_2 \overline{s}_1 \overline{s}_0 \\
e_3 &= \overline{s}_2 s_1 s_0 \\
e_2 &= \overline{s}_2 s_1 \overline{s}_0 \\
e_1 &= \overline{s}_2 \overline{s}_1 s_0 \\
e_0 &= \overline{s}_2 \overline{s}_1 \overline{s}_0
\end{aligned}
$$

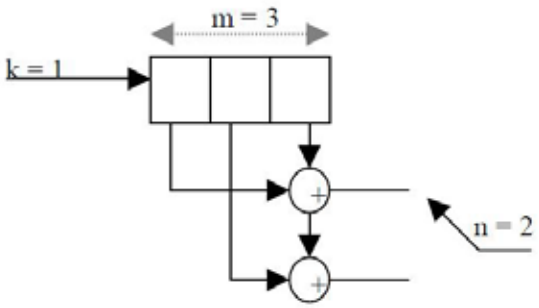


Figura 5

Sistema Propuesto como Codificador Convolucional

De esta manera, la constitución del codificador y decodificador fueron implementados a partir de la teoría y funciones anteriores.

Cálculos para el Codificador Convolucional (2,1,3)

Para el codificador implementado en el NIOS II se definió un sistema (2,1,3), es decir: un bit de la palabra de datos como entrada al codificador por cada corrimiento, dos bits de palabra de codificación por cada bit de palabra de datos y tres bits de longitud de registro, es decir, polinomios generadores de dimensión 3. En nuestro caso, la tasa es . El sistema codificador implementado se muestra esquemáticamente en la Figura 5.

La implementación se lleva a cabo a través del manejo del sistema operativo de tiempo real Qc-OS, de tal forma que las etapas de la mencionada codificación serán consideradas como tareas descritas en lenguaje *C*.

Estas tareas serán reguladas de la misma forma como se llevó a cabo en la etapa de codificación de bloques a través de semáforos, transferencia de datos mediante colas y activación de etapas por mailboxes.

El codificador/decodificador convolucional tiene las siguientes características:

- C(2,1,3) del cual se consideran tramas de 8 bits, una tasa de y un valor m=3.
- El tamaño de trama es esencial debido a la necesidad de acoplar el codificador de bloques considerado anteriormente con el codificador convolucional, ya que el primero presenta como salida un código de dicha longitud.
- Dada la tasa de generación, se plantea incluir dos generadores con las siguientes características: g1=[1 0 1] y g2=[1 1 0].

Por lo anterior, se genera un código de acuerdo con los elementos X-OR del registro de corrimiento, de tal forma que el sumador módulo 2 superior (de la Figura 5) genera el MSB del dibit codificado y por consiguiente el sumador inferior asume el papel de generador del bit LSB. Para programar la codificación convolucional se tomó en consideración el bosquejo funcional observado en la Figura 6.

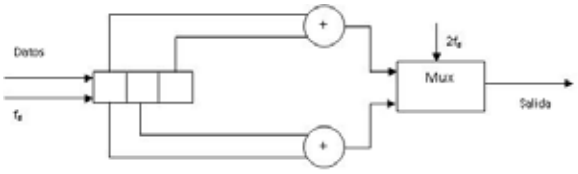


Figura 6

Bosquejo del Codificador Convolucional

III. Implementación de la Estación Base en el FPGA/NIOS II

En esta sección, se considera la construcción y operación del código correspondiente a cada bloque de la Figura 1. Asimismo, se da un panorama completo del funcionamiento de todo el sistema. Es importante remarcar que entre cada bloque se usan mailboxes, ya que son tareas que no pueden realizarse al mismo tiempo porque el resultado de cada tarea depende de la entrada de la tarea anterior. Además, se hace uso de un semáforo con el objeto de lograr la simulación de la llegada de tres mensajes al mismo tiempo y que debe leer dependiendo de quién haya tenido la primicia, esto considera además a la prioridad de las tareas. Para el envío de los mensajes de una tarea a otra en la ejecución de los demás procesos, se utilizan colas.

Main Section

En esta sección se hace la declaración de todos los elementos usados, los cuales son:

- 6 mailboxes
- 4 colas
- 1 semáforo
- 8 tareas

Lectura de Datos

La lectura de datos se hace a través de un semáforo. La primera vez que arranca el programa se dirige de forma directa a la Tarea 1, que tiene la prioridad más alta y de ahí comienza un semáforo para leer los demás mensajes. Esto se hace simulando que tres mensajes de 5 bits llegan al mismo tiempo, como en la realidad sucedería en una estación base. La introducción de los mensajes es a través de los switches, donde se tiene el siguiente formato:

- Mensaje 1: switches 0 al 4 con prioridad de 1.
- Mensaje 2: switches 5 al 9 con prioridad de 2, es decir, llega después del anterior.
- Mensaje 3: switches 10 al 14 con prioridad de 3, o sea, llega después del mensaje 2.

De acuerdo con las prioridades indicadas, el mensaje 1 tiene la prioridad más alta, seguida del mensaje 2 y el mensaje 3 la menor de los tres mensajes. Para ello, se crearon 3 tareas que reciben semáforos para leer cada uno de los grupos de interruptores. La dinámica de lectura es la misma: lee el mensaje de los switches dependiendo la prioridad, aplica una máscara para el caso de la recuperación de los mensajes 2 y 3 y los guarda en una cola para ser enviados a la siguiente tarea, que en este caso es el codificador de bloques.

Una vez que terminó con la lectura de los tres datos, o la ejecución de las tres tareas correspondientes a la lectura de cada uno de los tres datos, se envía un mailbox a la tarea que contiene el codificador de bloques.

Para que el programa pueda ejecutarse varias veces de manera automática, se agregaron dos instrucciones donde se detecta que un botón fue presionado. Si el botón 1 fue presionado, entonces comienza el proceso de codificación para todos los mensajes. Por medio del botón 3 se indica que se están introduciendo

los errores que anteriormente debieron haber sido puestos en los switches en las posiciones donde se requieren dichos errores, para después introducir los mensajes y presionar el botón 1 para que comience el proceso.

En cada una de las tareas se manda a llamar a la función mensajes(). Esta función se encarga de leer los datos desde los switches y envía ese dato por medio de una cola a la tarea que lo está solicitando, donde se le aplicará una máscara para obtener el dato deseado.

Codificador de Bloques

Como se mostró en la Figura 1, el primer bloque con el que comienza el sistema, es el codificador de bloques, que es por tanto a donde llegan los mensajes a enviar. Como se describió en la sección Main del programa, los tres mensajes son recibidos al mismo tiempo, son guardados en una cola y se envía un mailbox a la tarea correspondiente al codificador de bloques (Tarea 4) para indicarle que es su turno. De esta manera, dicha tarea recibe los tres mensajes en la cola y comienza a codificarlos uno por uno (esto se hace con un ciclo for), guardándolos de nuevo en esa misma cola para ser enviados a la Tarea 5, que corresponde al codificador convolucional, y finalmente se envía otro mailbox para indicarle que es su turno. El tamaño de cada mensaje recibido es de 5 bits, y el de cada mensaje codificado es de 8 bits.

Codificador Convolucional

El codificador en la implementación fue desarrollado en las tareas 5 y 6 del código general.

La tarea 5 tiene la función de recibir un conjunto de 3 tramas de 8 bits cada una proveniente del codificador de bloques de la etapa anterior y que corresponde a los mensajes codificados 1, 2 y 3 (originalmente de longitud 5). Se considera en dicha tarea el preámbulo a la codificación convolucional ya que genera las condiciones para llevar a cabo el ingreso de cada elemento al registro de corrimiento del codificador. Este registro de corrimiento se representa por las funciones del lenguaje *C* para el corrimiento de bits.

Es importante mencionar que la tarea 5 permanece inactiva en espera de un mailbox proveniente de la tarea 4, que corresponde a la última etapa del codificador de bloques. Una vez que el mailbox es recibido, se hace lectura de los mensajes que fueron proporcionados hasta esta tarea mediante el uso de una cola denominada CommQIngr1.

De forma similar, las tramas desplazadas en posición son proporcionadas a la tarea 6 para llevar a cabo la codificación. El medio para sincronizar el fin de la tarea 5 y el inicio de la tarea 6 es delegado a un mailbox (CommBox3).

La tarea 6, de la misma forma que la tarea 5, establece comunicación con la que le precedió y con la que sigue a través del manejo de la misma cola de 3 elementos de 16 bits. La activación corresponde al mailbox (CommBox3) esperado de la tarea anterior.

La tarea 6 presenta algunos elementos de importancia:

- Codificación.
    - Se realiza para cada mensaje de 16 bits (8 de datos y ceros

adicionales para llevar a cabo el corrimiento) la suma módulo 2 con cada generador.

- Se obtiene una trama al doble de dimensión (16 bits por cada 8 de datos) para cada uno de los 3 mensajes.
- La trama codificada corresponde al elemento que deberá transmitirse a través de un canal de comunicación (con una modulación respectiva) de tal forma que en caso de condiciones adversas, ésta será la que se vea afectada por el ruido.

Introducción de Errores

Para simular la transmisión y recepción de tramas de datos a través de un canal ruidoso, se toma lectura de los SW deslizables de la tarjeta DE2 con la finalidad de agregar bits erróneos a cada uno de los 3 mensajes. Los errores son proporcionados a través de una cola (CommQ).

Los bits erróneos son operados módulo 2 con los mensajes obtenidos del codificador.

Cabe mencionarse que los errores son proporcionados desde la tarea 1, ya que al inicio del proceso (antes del botón de arranque) es el momento en el cual se agregan o no los bits erróneos.

Con la finalidad de verificar los bits transmitidos por mensaje (16 bits codificados por el convolucional) se realizó la codificación a caracteres ASCII para cada bit de los mensajes, antes de la inclusión de errores y después de éstos. Cada caso de trama es colocado en una línea del mismo.

- Sin errores – línea 1.
- Con errores – línea 2.

Decodificador Convolucional

Finalmente, se activa a la tarea 7, que realiza la decodificación convolucional a través del mismo procedimiento utilizado en las tareas anteriores (manejo de cola y mailbox).

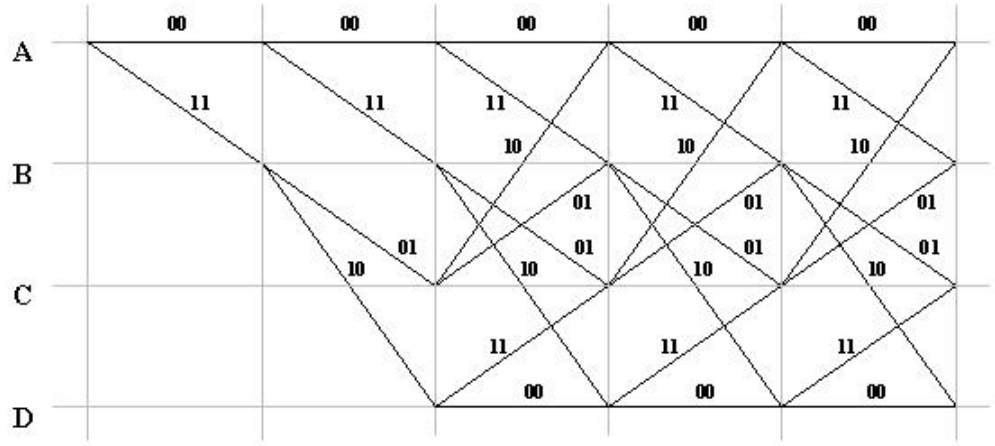


Figura 7

Trellis del Sistema

Para la implementación del decodificador se optó por emplear la técnica de trellis para llevar a cabo la recuperación de la información, así como el control de errores. La trellis utilizada para la implementación se muestra en la Figura 7.

En ella se observa el manejo de 4 estados a través de los cuales es posible codificar y decodificar un código convolucional.

La tarea 7 corresponde a todo el proceso de decodificación convolucional de la implementación. Con la finalidad de revisar el proceso, se dividirá en sus etapas más significativas.

- **Manejo de estado.**
    - Como se observó en la Figura 7, la trellis está constituida de trayectorias a través de 4 estados de operación que corresponden a los orígenes y llegadas de acuerdo a los códigos a codificar o a decodificar de un mensaje.
    - El estado final para cada decodificación de un dibit corresponde al estado inicial para la decodificación del dibit siguiente, por lo tanto deberá resguardarse.
    - Para llevar un correcto control de los estados, se consideró una trellis de estados siguientes, adicional a la trellis de códigos.
- **Ambigüedad en distancia de Hamming.**
    - En caso de que la distancia de Hamming sea igual para ambas opciones, se deberá esperar a conocer la distancia de Hamming del siguiente dibit, considerando como estado inicial las dos opciones posibles si el primer elemento fuera decodificado como 0 o como 1.
    - De las opciones, se escoge la de menor distancia de Hamming y en regresión de trayectoria se identifica al bit decodificado en el paso anterior.
- **Distancia de Hamming.**
    - En cada dibit a decodificar deberá conseguirse la distancia de Hamming para cada una de las dos opciones presentes dado un estado actual.
    - La distancia de Hamming es calculada del dibit a decodificar con respecto a las ramas que salen del estado en el que se encuentra.
    - Si la menor distancia corresponde a la rama superior, el bit decodificado es un 0.
    - Si la menor distancia corresponde a la rama inferior, el bit decodificado es un 1.

Finalmente los mensajes decodificados convolutivamente, pero aún codificados por bloques, se proporcionan al decodificador de bloques que corresponde a la tarea 8. La transferencia de los mensajes se lleva a cabo de la forma habitual como se llevó a cabo en las tareas anteriores.

Decodificador de Bloques

Para que este bloque comience a hacer su trabajo, recibe un mailbox de la Tarea 7 y una cola donde se tienen los 3 mensajes que pasaron por el decodificador convolucional y que finalmente entrarán al decodificador de bloques para

obtener los mensajes recibidos y compararlos con los enviados. El número de bits por cada mensaje es de 8 bits, y el mensaje de salida tendrá 3 bits. La programación de esta sección consta de las fórmulas indicadas en (14) y (16) para el cálculo del síndrome y el patrón de errores. El proceso es el siguiente: una vez que esta tarea recibe la cola con los 3 mensajes a decodificar, calcula el síndrome por medio de la función mostrada en (14) para después sumar el patrón de errores, cuya función es corregir los errores, y finalmente obtener el mensaje recibido tomando solamente los 5 bits menos significativos. Este resultado es mostrado en las salidas led y lleva a cabo una mejor comparación con los mensajes enviados a través de los switches. Asimismo, se muestra un letrero en los displays de 7 segmentos si hubo errores en la transmisión, tal como fue mencionado en secciones anteriores.

Después de este proceso, regresa a la tarea 1 para la lectura de mensajes y espera la señal de algún botón para que vuelva a comenzar.

IV. Probabilidad de Error

Uno de los parámetros más usados para el análisis de un sistema de comunicaciones, es la tasa de error de bit (BER), que es un registro histórico del verdadero rendimiento de error de bit de un sistema. Una tasa de error de bit se mide y luego se compara con la probabilidad de error esperada para evaluar el rendimiento de un sistema.

La fórmula general para calcular la BER, se muestra en la siguiente ecuación (17).

$$BER(x) = \frac{P_t}{2}\left[erfc\left(\frac{x}{\sqrt{2\sigma}}\right) + erfc\left(\frac{1-x}{\sqrt{2\sigma}}\right)\right]$$

Donde:
P_t es la potencia de transmisión.
es la Varianza.
^{2} es la Desviación estándar.
t es el Tiempo.

Sin embargo, también existen fórmulas ya establecidas para el cálculo de la tasa de error de bit, dependiendo del tipo de codificador a usar. El cálculo de BER para un codificador de bloques se realiza mediante la ecuación (18).

$$P_{BCB} \approx \frac{1}{n}\sum_{j=t+1}^{n} j\binom{n}{j} p^j (1-p)^{n-j}$$

$$d_{\min} \le \frac{n\left(2^{k-1}\right)}{2^k - 1}$$

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

Donde:
P_{BCB} es la Probabilidad de error en bit para un codificador de bloques.
k es el número de bits del mensaje a enviar.

n es el número de bits por mensaje codificado.
t es la capacidad de corrección de errores.
d_{min} es la distancia mínima entre errores contiguos.
p es la probabilidad de transición.

Para un código convolucional, la relación está dada por la expresión (21).

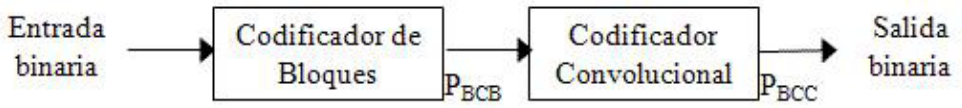
$$P_{BCC} \leq \left. \frac{dT(D,N)}{dN} \right|_{N=1, D=2\sqrt{p(1-p)}}$$

Donde:
P_{BCC} es la Probabilidad de error en bit para un codificador convolucional.
T es la Capacidad de corrección de errores.

En el diagrama de la Figura 9 se muestra la ubicación de estas probabilidades en un sistema.

Figura 8

Diagrama con Ubicación de Probabilidades de Error



De manera particular para este trabajo, la fórmula de probabilidad de error en bit para el codificador de bloques quedaría de la forma como se representa en (24).

$$d_{\min} \leq \frac{n\left(2^{k-1}\right)}{2^k - 1} = 4.5714$$

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = 1$$

$$P_{BCB} \approx \frac{1}{8} \sum_{j=2}^{8} j \binom{8}{j} p^j (1-p)^{8-j}$$

Para el caso del codificador convolucional, y basado en el diagrama de la Figura 3, se obtienen las siguientes ecuaciones (expresiones 25 a 28), de las cuales se obtiene la función de transferencia mostrada en (29).

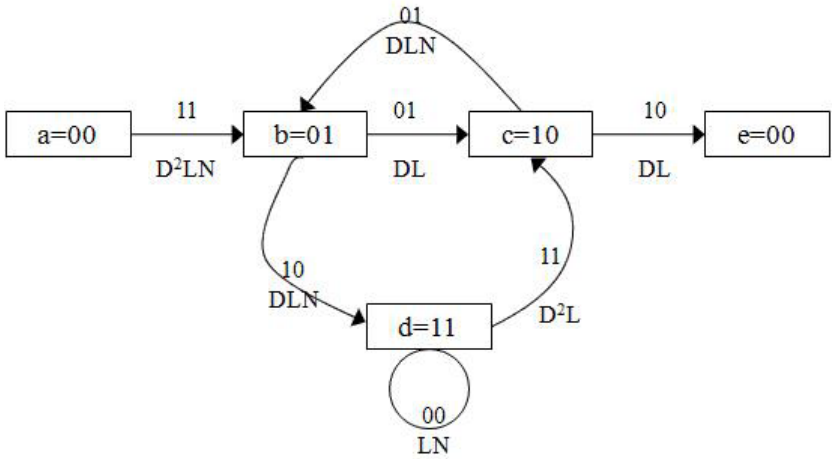


Figura 9

Diagrama de Estados.

$$X_b = D^2 LNX_a + DLNX_c$$
$$X_c = DLX_b + D^2 LX_d$$
$$X_d = DLNX_b + LNX_d$$
$$X_e = DLX_c$$

$$T(D) = \frac{D^7}{\left(1 - D - D^3\right)^2} + \frac{D^4}{\left(1 - D - D^3\right)}$$

Por tanto, a partir de la fórmula (25), la probabilidad de error de bit P_B está dada por la expresión 30 y 31:

$$P_{BCC} \leq \frac{dT}{dN}\left(\frac{D^4 L^3 N}{1-(DLN + D^3 L^2 N}\right)_{N=1, D=2\sqrt{p(1-p)}}$$

$$P_{BCC} \leq \frac{\sqrt{p(1-p)}^7}{\left(1-\sqrt{p(1-p)}-\sqrt{p(1-p)}^3\right)^2} + \left(\frac{\sqrt{p(1-p)}^4}{1-\sqrt{p(1-p)}-\sqrt{p(1-p)}^3}\right)$$

El cálculo del error de bit del sistema se determina de la siguiente manera: dado que se tiene un codificador de bloques (8,5) como código externo y un codificador convolucional como código interno. El primero tiene como parámetro d_{min} = 4, y el segundo tiene una distancia d_f = 5. Por tanto, el código concatenado tiene una mínima distancia efectiva de 20. Por tanto, hay 20 palabras-código posibles. La probabilidad de error en símbolo en el decodificador de bloques puede ser obtenido usando la ecuación (30), y después se obtiene el comportamiento del código externo usando (28), y por tanto, el comportamiento del código concatenado es obtenido usando la probabilidad de error en conjunto con la función de transferencia del código convolucional.

Conclusiones

Los sistemas de corrección de errores son grandemente utilizados en los sistemas de comunicación de alta tasa debido a que permiten la reducción de retransmisiones de amplios volúmenes de datos, lo que radica en un mejor rendimiento y aprovechamiento de los recursos.

Aunque el objetivo plateando en el desarrollo de este trabajo se cumplió en su totalidad, es posible tomar los resultados como fundamentos para el desarrollo de nuevos sistemas de comunicaciones a través de los cuales se puedan desarrollar pruebas de rendimiento al implementar un algoritmo de medición, cálculo y comparación de la BER.

Debido a que el sistema tiene la capacidad de expansión de longitud de trama, así como de subsistemas para la cadena de comunicación, se pretende trabajar en dos rubros de gran importancia. Para el primer caso, es posible adoptar una longitud de trama distinta a la utilizada, ya que para fines didácticos el codificador de bloques (5,8) y el codificador convolucional de ½ no presentan

problemas, pero en aplicaciones reales se requiere de mayores capacidades. Los tamaños de los codificadores dependerán directamente de la adopción de un estándar de comunicación, tal como GSM, WiMax u otro. Parte del trabajo a futuro corresponde en identificar el estándar que brinde las condiciones más adecuadas para el análisis en cuestión. Esto implicará adaptar el tamaño de datos para formar las tramas con características que indique el estándar elegido. En cuanto a la expansión de subsistemas para la cadena de comunicaciones, se pretende incluir un mayor número de elementos, los cuales en número y tipo dependerán directamente del estándar elegido para el análisis, aunque es posible hablar de la construcción de un modulador/demodulador digital embebido, ya que éste se requiere para cualquier estándar.

Autores

Jaime Humberto Pech Carmona. Es Ingeniero en Electrónica, graduado del Tecnológico de Estudios Superiores de Ecatepec en 2001. Obtuvo el grado de Maestro en Ciencias en el Centro de Investigación en Computación (CIC) del Instituto Politécnico Nacional en 2006. En ese periodo realizó investigación sobre procesamiento de señales y reconocimiento de voz y locutores. En 2012 obtuvo el grado de Doctor en Ciencias de Ingeniería del ITESM. De 2013 a la fecha, ha realizado investigación en el área de comunicaciones y desvanecimiento de señal. Actualmente forma parte de la División de Ingeniería Electrónica del Tecnológico de Estudios Superiores de Ecatepec.

Karla Vázquez Ascención. Es Licenciada en Informática, graduada del Tecnológico de Estudios Superiores de Ecatepec en 2004. Obtuvo el grado de Maestra en Tecnología Educativa por el Tecnológico de Monterrey en 2011. Desde 2012 ha conducido investigaciones sobre la aplicación de tecnologías de la información en la educación. Actualmente trabaja con la investigación de procesos estocásticos dentro de la División de Ingeniería en Sistemas Computacionales del Tecnológico de Estudios Superiores de Ecatepec.

Bibliografía

[1] Sklar, Bernard. Harris, Frederic. "The ABCs of Linear Block Codes". *IEEE Signal Processing Magazine*. Julio 2004.

[2] Sklar, Bernard. *Digital Communications: Fundamentals and Applications.* Pearson Education. 2010.

[3] Proakis, John. *Digital Communications.* Mc. Graw Hill. 1995.

[4] Ruiz, Jairo A. *Codificación Convolucional.* Universidad Distrital Francisco José de Caldas, Bogotá Colombia. http://www.udistrital.edu.co/comunidad/profesores/jruiz/jairocd/

[5] Carlson, A.B., Crilly, P.B. y Ruttedge, J.C., *Communications Systems.* Págs 573-589, 4ª ed. 2002.

[6] Sandoval, Cecilia. "Modular Design of scheme coding concatenated for correction error with programming of hardware". *Revista Chilena de Ingeniería*. Vol. 16 No. 2. 2008.

[7] Murray, Leslie. *Introducción a los códigos convolucionales.* Escuela de Ingeniería Electrónica, Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional del Rosario.

[8] Proakis, John. *Fundamentals and Communication Systems.* Global Edition. Pearson. 2015.

[9] Halsall, Fred. *Computer Networking and the Internet.* Addison Wesley. 5th edition 2005.

[10] Haykin, Simon. *Sistemas de comunicaciones.* Limusa. 2002.