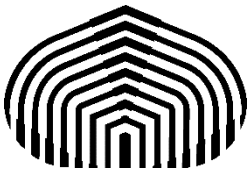


UNIVERSIDAD SIMON BOLIVAR
Departamento de Electrónica y Circuitos
Comunicaciones II
(EC-3423)

Práctica 5

Códigos de convolución

Elaborador por:

José Morán - 1410714

Adrián González – 1410433

Preparación

En el codificador convolucional de la figura 1, los bits de la secuencia de entrada se procesan uno a uno. Se pide que:

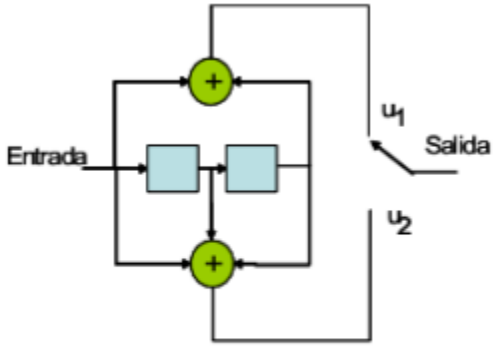
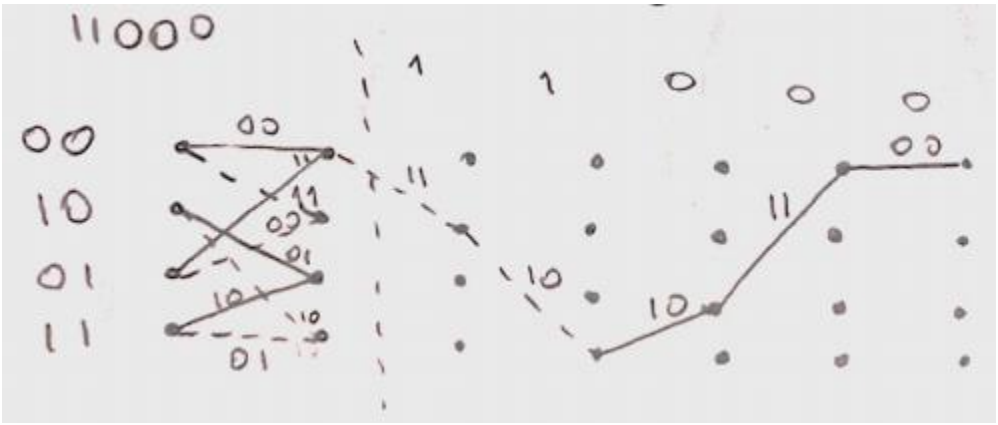


Figura 1

a) **Determine los parámetros del código (constraint length, tasa del código, número de estados).**
   - Constraint length: K = 3
   - Tasa del código: R = ½
   - Numero de estados: 4

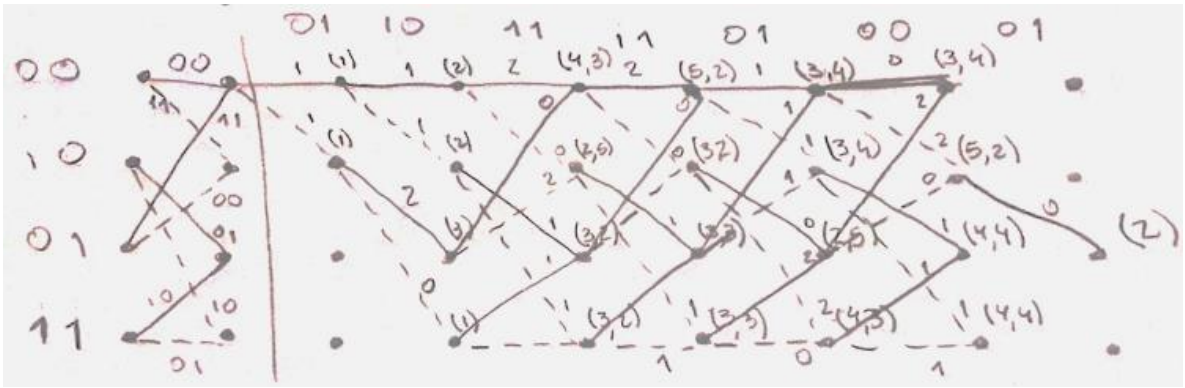
b) **Codifique usando el diagrama de Trellis la secuencia 11000**



El mensaje codificado es 1110101100

c) **Considere que se recibe la secuencia**

d) **01101111010001, decodifíquela usando el diagrama de Trellis y señale cuantos errores hay en la secuencia recibida.**

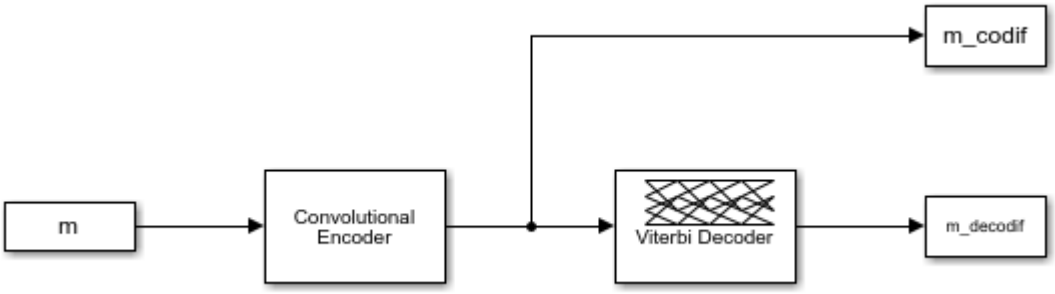


El mensaje decodificado es 1001010 y la cantidad de errores corregidos es 2.

e) Escriba un script en Matlab para simular el codificador y el decodificador.

```
%% Codificador convolucional

%Preliminares
clear;clc;

%Definiciones
const_len = 3;       %Constraint length
code_rate = 1/2;     %Tasa del codigo
disp('Mensaje original');
m = [1 1 0 0 0];      %Mensjae original
disp(m);

%Ecuaciones de paridad
u1 = str2double(dec2base(bin2dec('101'),8));
u2 = str2double(dec2base(bin2dec('111'),8));
code_gen = [u1 u2];

%Generador de la estructura Trellis
t = poly2trellis(const_len,code_gen);

%Codificador del mensaje
disp('Mensaje codificado');
m_codif = convenc(m,t);
disp(m_codif);

%% Decodificador por medio del algoritmo de Viterbi
disp('El mensaje decodificado es');
m_decodif = vitdec(m_codif,t,const_len,'trunc','hard');
disp(m_decodif);
```

f) Diseñe el modelo de Simulink equivalente al script del numeral e.



Laboratorio

1. **Compruebe el funcionamiento del sistema implementado en el punto 2.e de la preparación, codificando y decodificando la entrada 11000.**

Al ejecutar el script, se obtiene lo siguiente en la consola:

```
Mensaje original
     1     1     0     0     0

Mensaje codificado
     1     1     1     0     1     0     1     1     0     0

El mensaje decodificado es
     1     1     0     0     0
```

Para probar la decodificación, se realiza una prueba con la secuencia de la parte d. El resultado es el siguiente:

```
La secuencia recibida es
     0     1     1     0     1     1     1     1     0     1     0     0     0     1

El mensaje decodificado es
     1     0     0     1     0     1     0
```

Se observa que se obtuvo el mismo resultado.

2. **Compruebe el funcionamiento del sistema implementado en el punto 2.f de la preparación, codificando y decodificando la entrada 11000.**

Para la simulación se redactó el siguiente script:

```
%% Simulink

%Preliminares
clear;clc;

%Definiciones
const_len = 3;      %Constraint length
disp('Mensaje original');
m = [1 1 0 0 0];     %Mensjae original
disp(m);

%Ecuaciones de paridad
u1 = str2double(dec2base(bin2dec('101'),8));
u2 = str2double(dec2base(bin2dec('111'),8));
code_gen = [u1 u2];
%Generador de la estructura Trellis
t = poly2trellis(const_len,code_gen);

%Simulacion
sim('conv_coder');
disp('El mensaje codificado es')
disp(m_codif(1,:));
disp('El mensaje decodificado es')
disp(m_decodif(1,:));
```

Al ejecutarlo, se obtiene lo siguiente:

```
Mensaje original
     1     1     0     0     0

El mensaje codificado es
     1     1     0     1     1     1     0     0

El mensaje decodificado es
     1     0     0     0
```

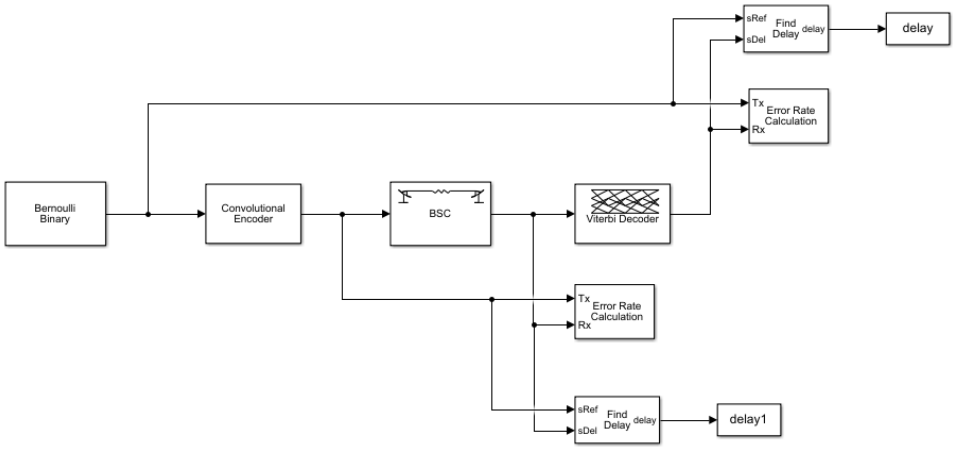
Claramente hay un error ya que, en primer lugar, el mensaje decodificado no coincide con el original y tiene un bit menos, además el mensaje codificado tiene 8 bits y debería tener 10 y no coincide con el resultado obtenido anteriormente. Por ensayo y error, se determinó que agregando un 0 al comienzo del mensaje se puede obtener codificación correcta, como se muestra a continuación

```
Mensaje original
     0     1     1     0     0     0

El mensaje codificado es
     1     1     1     0     1     0     1     1     0     0

El mensaje decodificado es
     1     1     0     0     0
```

Tal parece que el bloque Convolutional Encoder consume el primer bit del mensaje original para sincronizarse y luego codifica correctamente.

3. **Simule continuamente el sistema 2.f con una entrada aleatoria, añadiendo errores binarios. Calcule la tasa de error usando el bloque Error Rate Calculation.**

Para esta parte, se modifico el diagrama de bloques en Simulink como se muestra.



La secuencia arbitraria de bits se genera en el bloque Bernoulli Binary sequence, que fue configurado para que generara unos y ceros con la misma probabilidad. El bloque BSC introduce errores con un 1% de probabilidad. Los bloques Find Delay se introducen para verificar si hay retraso en las secuencias de bits original y decodificada, y antes y después de la introducción del error. Se obtiene que hay un retraso de 3 muestras entre la secuencia transmitida y la decodificada y se ajusto el bloque Error Rate Calculation adecuadamente. La simulación se detiene cuando se han introducido 1000 errores.

Los bloques Error Rate Calculation arrojan que de los 1000 errores que se produjeron, el decodificador pudo corregir 488 (512 no se corrigieron) errores cuando se generaron 4895 bits. Esto conlleva a una probabilidad de error igual al 10.46% con solo codificar y decodificar, pudiendo reducirse mas si se implementa un algoritmo de corrección de errores junto a la codificación de convolución.