CAPÍTULO 4

Simulación de un ADSL *transceiver* en LabVIEW

En este capítulo se incluye el trabajo principal de esta tesis referente a la simulación del módem ADSL o ADSL *transceiver* en LabVIEW. El capítulo comienza con una breve introducción a LabVIEW y su ambiente de diseño. Posteriormente se explica cómo se modelan y programan los diferentes bloques del módem ADSL para realizar la simulación. Estos bloques fueron discutidos detalladamente en el capítulo 2. Se incluyen imágenes de algunos de los códigos programados con el lenguaje gráfico de LabVIEW.

4.1 INTRODUCCIÓN A LABVIEW

LabVIEW es la abreviación para ***Lab****oratory* ***V****irtual* ***I****nstrument* ***E****ngineering* ***W****orkbench*. Es una poderosa y flexible herramienta para la adquisición, análisis y presentación de datos que puede calificarse como un ambiente excelente para aplicaciones de instrumentación y control de procesos. En este proyecto se utilizan la versión profesional 7.1 y un manual editado por *National Instruments* para la versión estudiantil 7.0 [BIS04]. Este software nos permite desarrollar, entre otras cosas, instrumentos virtuales con una interfase gráfica muy amigable, permitiendo que cualquier persona con conocimientos en la materia para la cual se desarrolla el instrumento, pueda interactuar y al mismo tiempo cambiar y/o ajustar los campos variables en el instrumento como si éste existiera físicamente.

Detrás del instrumento virtual, existe la interfase de programación o diagrama a bloques, donde se hace uso de un lenguaje de programación gráfico o lenguaje G. Cada elemento de entrada o salida en la interfase del usuario o panel frontal tiene su correspondiente representación en el diagrama a bloques. El usuario, como se ha mencionado con anterioridad, puede cambiar los valores de entrada en cualquier momento durante el tiempo de ejecución del programa. El lenguaje G nos permite "programar" las aplicaciones mediante la interconexión lógica entre diversos elementos o instrumentos virtuales. Esta forma de programación también incluye características similares a las de otros lenguajes de programación como C, es decir, es capaz de aceptar variables locales o globales, utilizar opciones de control de lazos y flujo, llamado a funciones y algunas otras características muy funcionales.

En LabVIEW a los programas se les llama Instrumentos Virtuales o simplemente VI's, éstos se desarrollan con instrumentos más pequeños incluidos en muchas librerías con las que LabVIEW cuenta y que pueden utilizarse para realizar diferentes tipos de programas de acuerdo a las necesidades del usuario. Estas librerías cubren desde operaciones matemáticas básicas hasta funciones avanzadas de procesamiento digital de señales, manejo de arreglos y matrices, etc.

Una de las características más novedosas y funcionales de este software es su capacidad para aceptar y ejecutar *scripts* de MATLAB, lo que hace aún más grande el alcance de esta herramienta y es por ello que se decidió realizar esta simulación en LabVIEW.

La Figura 4.1 muestra un ejemplo del panel frontal de un instrumento virtual que se incluye en la sección de ejemplos de LabVIEW. Este ejemplo corresponde a la simulación de un sistema de control y monitoreo de los niveles de agua en un tanque. Como se puede apreciar, la interfase de usuario es muy amigable ya que nos permite modificar las variables globales como si fueran botones de un instrumento real. También se pueden mostrar los datos y resultados en forma gráfica ya sea en tiempo real o de manera simulada. La Figura 4.2 muestra el diagrama a bloques del ejemplo en discusión y como se puede observar constituye la parte de programación mediante el uso del lenguaje G.

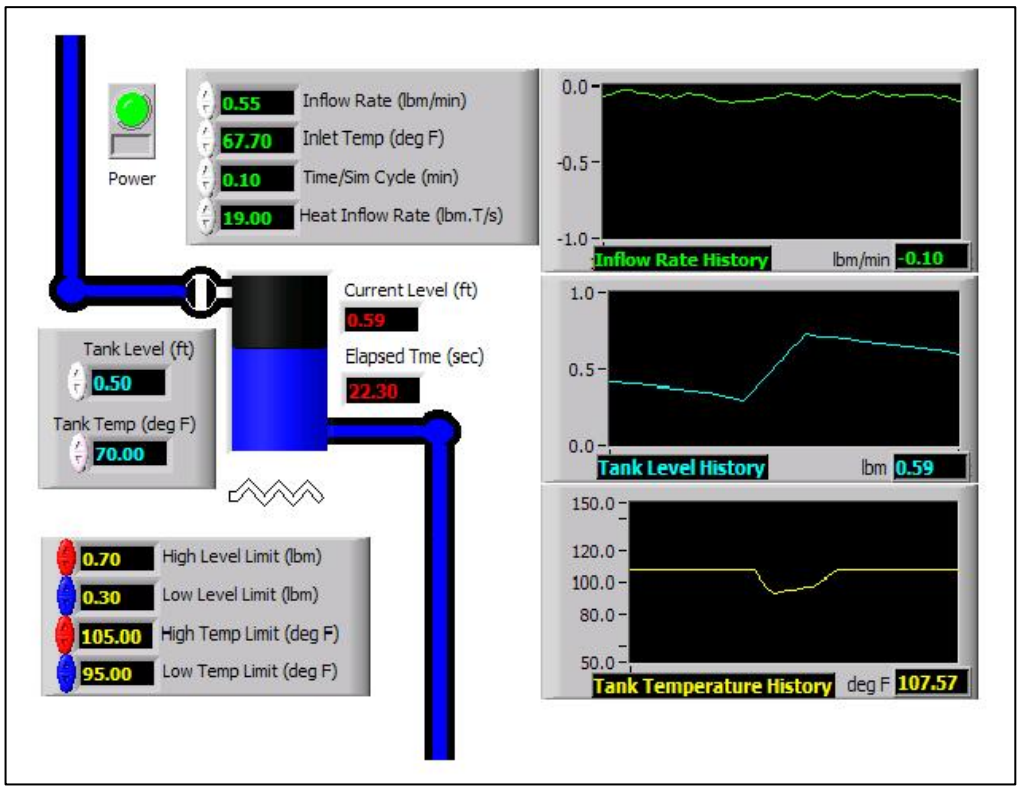

Figura 4.1 Ejemplo de panel frontal de un Instrumento Virtual en LabVIEW.

El diagrama a bloques de la Figura 4.2 muestra claramente la interconexión de varios bloques funcionales dispuestos de tal forma que en conjunto realizan una tarea específica. Se pueden agrupar varias de estas tareas para crear de esta forma sub-instrumentos virtuales (SubVI's) que a su vez dan paso al instrumento virtual principal. Con lo anterior, podemos concluir que el diseño de un instrumento virtual puede realizarse con característica de *top level*, donde los sub-instrumentos virtuales pueden ser llamados desde el instrumento virtual principal.

Los operadores de control de lazos y flujo son muy importantes en el diseño de los programas en cualquier lenguaje de programación y en LabVIEW no es la excepción. Estas características son muy útiles y funcionales pues nos permiten controlar nuestro programa a nuestra conveniencia. En este ejemplo el lazo color gris representa un controlador de flujo condicional, equivalente a un *while* en cualquier lenguaje de programación de alto nivel como C, Visual Basic, etc. Las líneas que conectan cada bloque representan los pasos de

valores y variables, y funcionan como en otros ambientes de simulación tales como *Simulink*.

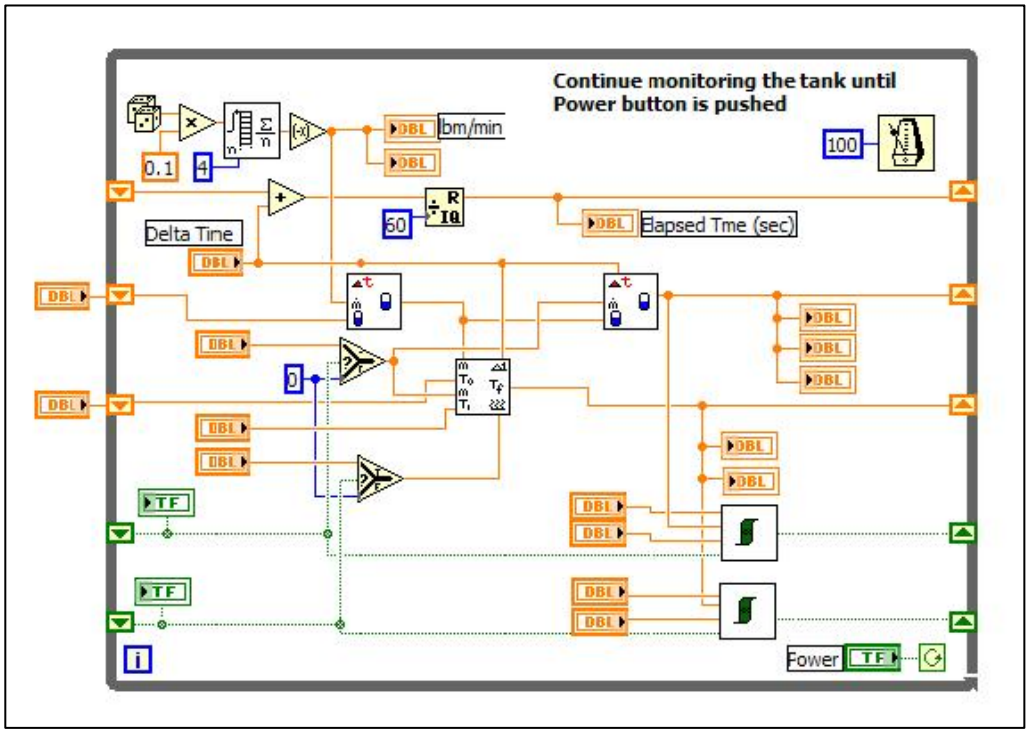

Figura 4.2 Diagrama a bloques del ejemplo de niveles de agua en un tanque.

4.2 SIMULACIÓN DEL TRANSMISOR DMT

En este proyecto de tesis se modela y simula la forma en que se trasmiten los datos a través de cada uno de los bloques que conforman el módem o *transceiver* ADSL. La idea principal es calcular el BER (*Bit Error Rate),* por lo tanto se decidió no realizar los bloques de conversión análoga digital y viceversa para simplificar el modelado y la simulación misma, ya que no existe noción alguna del tiempo. La programación y simulación de esta parte se sugiere en la sección de trabajos a futuro.

El BER es la forma más común de medir o contar los errores de ejecución en un circuito que transmite datos y se define como la probabilidad de que un bit transmitido sea recibido en error. Este parámetro es de gran importancia para medir el desempeño de un sistema digital de comunicaciones. El BER se puede calcular de la siguiente manera:

$$BER = \frac{Número\ de\ errores}{Número\ total\ de\ bits} \tag{4.1}$$

La simulación del módem ADSL se realiza en LabVIEW 7.1 Profesional con la ayuda de MATLAB 7.0 R14. Se utilizaron dos *toolbox* de MATLAB encontrados en Internet bajo licencia gratuita, uno de ellos se utilizó para implementar el ecualizador en el dominio del tiempo (*DMTTEQ Toolbox*) [EVA05], en tanto que el otro es una *toolbox* para simular un sistema ADSL (*ADSLToolbox for MATLAB*) [BAC01].

Los conceptos de conversión serial a paralelo y paralelo a serial son triviales al tratar de implementarlos en LabVIEW. En un S/P, por ejemplo, una larga serie de bits se divide en varias sub-series de igual tamaño que pueden ser procesadas al mismo tiempo. En un P/S estas sub-series se unen unas tras otras para formar nuevamente la serie larga de bits. Sin embargo, para su programación solo basta con manipular un vector en varias columnas de una matriz, o tomar una matriz y realizar una concatenación de sus columnas.

En ADSL se necesita que estas sub-series resultantes de la conversión serial a paralelo sean de diferentes tamaños, lo que resulta difícil de manipular si pensamos en vectores y matrices. Se necesitaría manipular el vector de entrada de acuerdo a la tabla de asignación de bits para crear 256 vectores diferentes con tamaños diferentes (un vector para cada sub-canal o tono DMT), que alimenten el banco de codificadores QAM en forma paralela. Esto lógicamente no resulta práctico, pero tampoco se puede hacer con una matriz con varias columnas porque ésta solo maneja dimensiones únicas. Por lo tanto, la generación aleatoria de bits de entrada se realiza independientemente para cada sub-canal dentro del banco QAM, de acuerdo con la tabla de asignación de bits.

La tabla de asignación de bits se realiza de manera aleatoria basándonos en la tabla de referencia que se muestra en la Tabla 4.1 [LEV00], esto debido a que en este trabajo no se implementa el período de inicialización y prueba descrito en el capítulo 2, que es necesario para calcular la relación señal a ruido (SNR) que se utiliza para determinar el número de bits que puede transportar cada sub-canal. El algoritmo base que se utiliza para

determinar la tabla de bits fue tomado de [BAC01] y se realizaron los cambios necesarios para nuestras necesidades. El instrumento virtual resultante se muestra en la Figura 4.3, donde se incluye el panel frontal y su correspondiente diagrama a bloques.

Tabla 4.1 Tabla SNR de referencia [LEV00].

Tabla SNR de Referencia	
BITS	**SNR_{ref}**
2	14
3	19
4	21
5	24
6	27
7	30
8	33
9	36
10	39
11	42
12	45
13	48
14	51
15	54

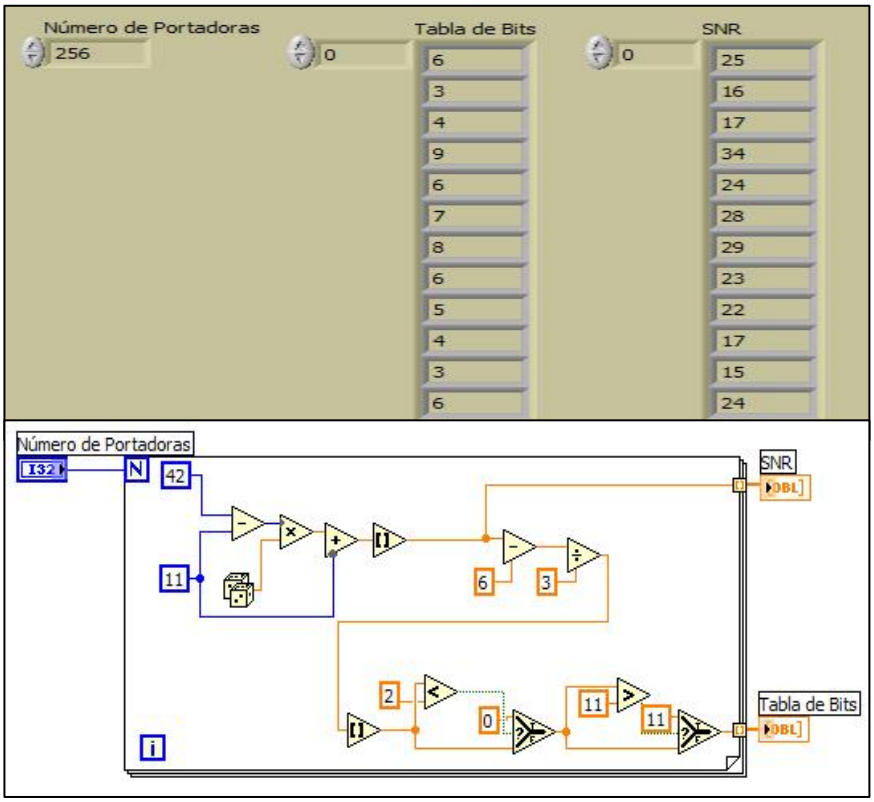

Figura 4.3 Instrumento virtual de la tabla de asignación de bits.

El banco de codificadores QAM se desarrolló haciendo uso de la capacidad que tiene LabVIEW para aceptar y ejecutar *scripts* de MATLAB. Se utilizó la función *qammod* del *Toolbox* de Comunicaciones que incluye la versión 7.0 de MATLAB. Este banco QAM también puede implementarse con la librería de modulación de *National Instrument* para LabVIEW, sin embargo no se implementó con esta librería porque no se tuvo acceso a ella.

La función *qammod* acepta dos argumentos, uno es la serie de *b* bits correspondiente a cada sub-canal convertida a número decimal y el otro es el tamaño de la constelación ($M=2^b$). Por razones del tiempo de ejecución de MATLAB, esta simulación solo acepta hasta un máximo de 11 bits por cada sub-canal. La razón principal de esta decisión es que una constelación de 32768 puntos (correspondiente al máximo de 15 bits por sub-canal en ADSL) requiere mucho tiempo de procesamiento por parte del servidor *script* de MATLAB. La forma como se implementó el banco de 256 codificadores QAM fue a través de un lazo *for* que permite ejecutar la función *qammod* 256 veces equivalentes a los 256 codificadores requeridos.

En la Figura 4.4 se observa el algoritmo desarrollado en MATLAB para el banco de codificadores QAM, donde *k* es el número de bits para cada sub-canal, *M* es el tamaño de la constelación, *x* es la serie de bits generados aleatoriamente para cada sub-canal, *xsym* es *x* representada en número decimal y la variable *y* de salida es un vector de números complejos de tamaño N=256.

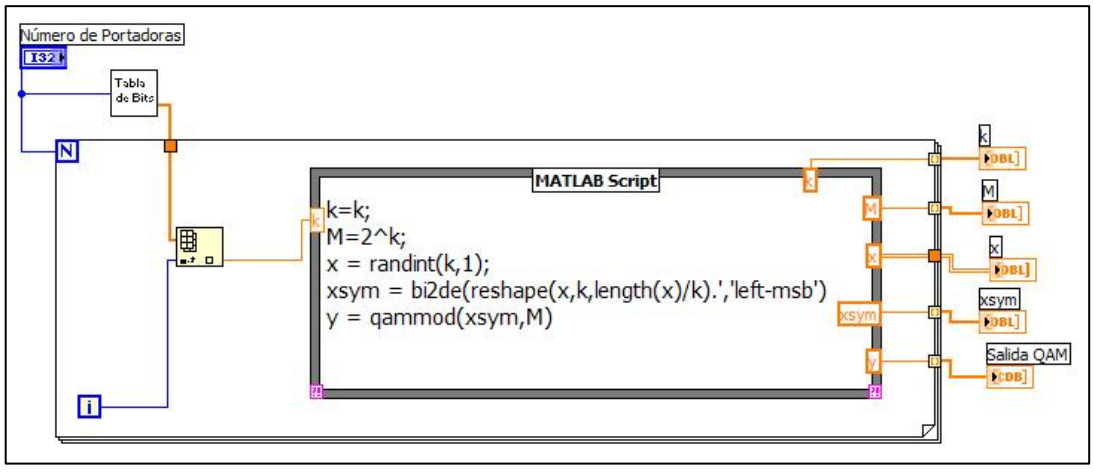

Figura 4.4 Instrumento virtual del banco de codificadores QAM.

El siguiente bloque correspondiente al espejo de datos se muestra en la Figura 4.5, cuya principal tarea es obtener el complejo conjugado de las N muestras del banco QAM para formar un bloque de 2N muestras. El instrumento virtual que se muestra realiza la siguiente función [ERW02]:

$$X_i = \begin{cases} X_i & i = 1, \cdots, N-1 \\ \mathrm{Re}\left(X_N\right) & i = 0 \\ \mathrm{Im}\left(X_N\right) & i = N \\ X^*_{2N-i} & i = N+1, \cdots, 2N-1 \end{cases} \qquad (4.2)$$

donde X_i es el vector con N sub-símbolos QAM.

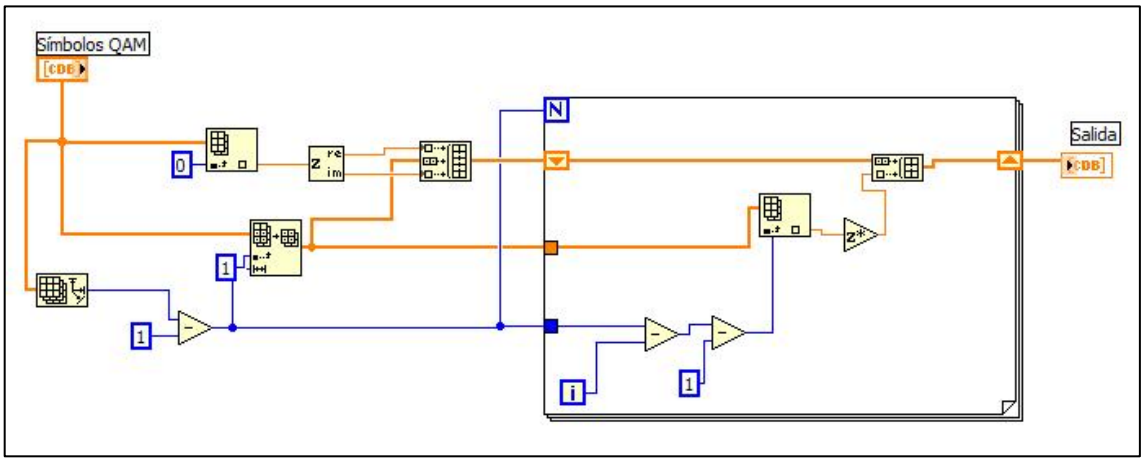

Figura 4.5 Instrumento virtual del espejo de datos.

La Transformada Rápida de Fourier Inversa (IFFT) está incluida en las librerías de LabVIEW, por lo que su desarrollo consistió en configurar el instrumento virtual IFFT.vi a las necesidades de nuestro sistema. La función de este bloque es realizar un mapeo de los sub-símbolos QAM a las sub-portadoras, es decir, a las diferentes frecuencias disponibles y de esta forma producir un símbolo DMT de 2N muestras en el dominio del tiempo. Este proceso también es necesario para la simulación del convertidor análogo digital, puesto que las muestras obtenidas del bloque IFFT son de valor real. Como se mencionó anteriormente, la IFFT se utiliza en este tipo de procesamiento porque es una de las formas o algoritmos más efectivos para realizar la Transformada Discreta de Fourier Inversa (IDFT).

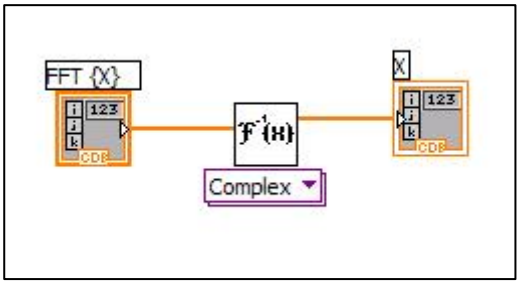

Figura 4.6 Instrumento virtual de IFFT.

La adición del prefijo cíclico se implementó tomando las últimas *v* muestras del bloque IFFT. Para ello basta obtener un sub-arreglo de tamaño *v* del arreglo principal de 2N muestras y agregarlo a éste último. En LabVIEW estas operaciones vienen incluidas en la librería de arreglos. Este bloque permite la elección del tamaño de prefijo, entre valores de 8, 16, 32 y 64, que son los más utilizados. En ADSL el estándar considera el tamaño *v*=32, por lo tanto el bloque de salida tendrá 2N+*v* muestras que es igual a 2(256)+32=544 muestras. Esta elección se logra con un lazo de condición, en este caso un lazo *case* que realiza tareas equivalentes al *if* y *case* del lenguaje C. La Figura 4.7 muestra este programa o instrumento virtual.

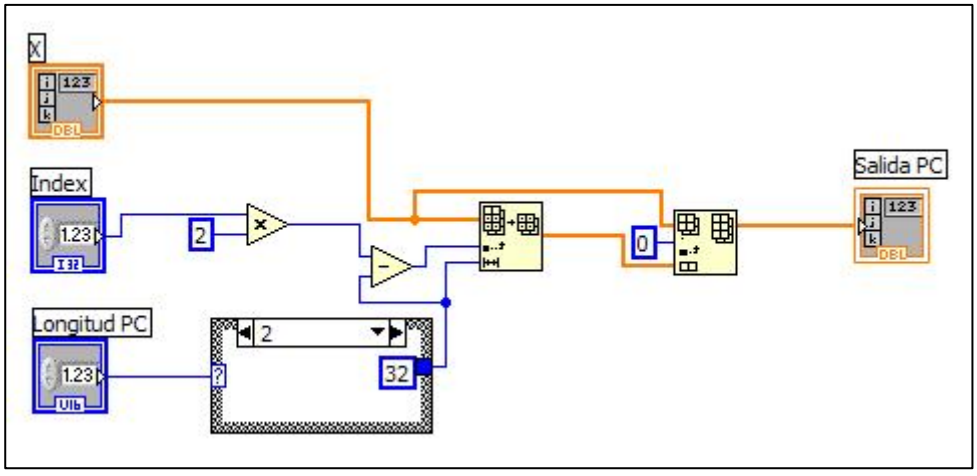

Figura 4.7 Instrumento virtual de adición de prefijo cíclico.

4.3 SIMULACIÓN DEL CANAL DE TRANSMISIÓN

El canal de transmisión de un sistema ADSL es la línea telefónica existente entre el usuario y la oficina central. Este cable de cobre puede ser de muchas formas, ya que éstos pueden

ser de diferentes longitudes y calibres, y pueden contener algunos inconvenientes mecánicos como puntos de unión y puentes (*bridged taps*). Por lo tanto, una línea telefónica o lazo de abonado puede modelarse de muchas maneras de acuerdo a las condiciones particulares de cada línea. Sin embargo, existen modelos de canal estandarizados que se utilizan para realizar pruebas y simulaciones a los sistemas ADSL. Estos modelos se conocen como *CSA Loops* (*Carrier Service Area Loops)* y tienen varias configuraciones de puentes, calibres y longitudes (ver Figura 4.8).

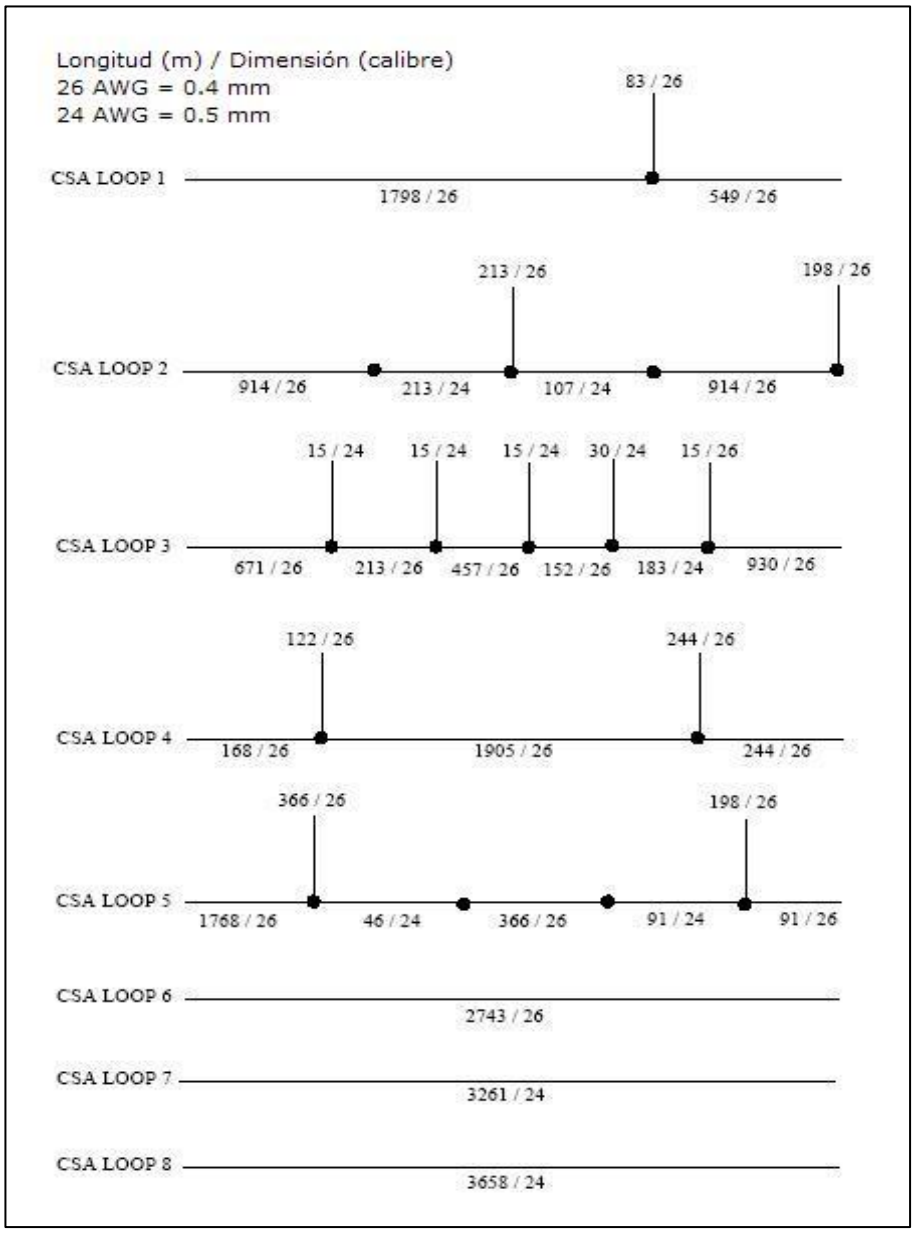

Figura 4.8 Modelos de canal estandarizados para pruebas [BAC01].

Cada una de estas configuraciones de CSA tiene respuesta al impulso diferente, es decir, cada una de las respuestas al impulso consiste de 512 muestras diferentes. Es por esto que cada uno de los modelos del canal de transmisión puede implementarse con un filtro de Respuesta al Impulso Finita (FIR) de 512 *taps* (longitud). . La ecuación (4.3) representa la convolución en tiempo discreto que se realiza para obtener la salida de un filtro digital FIR

$$y[k] = \sum_{m=o}^{N-1} h[m] \cdot x[k-m] = h[n] * x[n] \qquad (4.3)$$

donde k = 0, 1, 2,…, N-1, $h[n]$ es la respuesta al impulso del canal, $x[n]$ es la secuencia de entrada, $y[k]$ es la secuencia de salida y n es el número de *taps* o coeficientes en el filtro.

En la Figura 4.9 se muestra un diagrama a bloques de un filtro FIR con N *taps*.

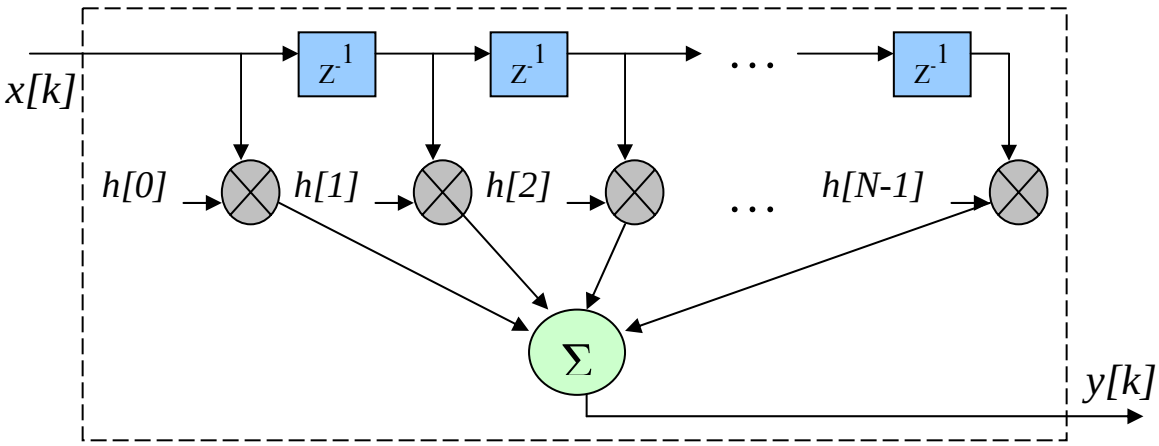

Figura 4.9 Filtro FIR con N *taps* [ARS99]

Los 512 coeficientes para este filtro FIR se obtuvieron del *toolbox DMTTEQ (Discrete Multitone Time-domain Equalizer)* para MATLAB [EVA05]. Este *toolbox* incluye los ocho modelos de canal descritos anteriormente y ejecuta una función que calcula los coeficientes necesarios para realizar el filtro FIR. Sin embargo, los coeficientes no los regresa al programa principal y por lo tanto se implementó un instrumento virtual que regresa los valores de los coeficientes. La Figura 4.10 muestra el panel frontal y el diagrama a bloques de esta etapa. Este instrumento virtual también calcula los coeficientes del ecualizador en el dominio del tiempo (TEQ), que puede modelarse de igual manera con

un filtro FIR de 16 *taps*. Este bloque se detalla en la sección referente a la simulación del receptor.

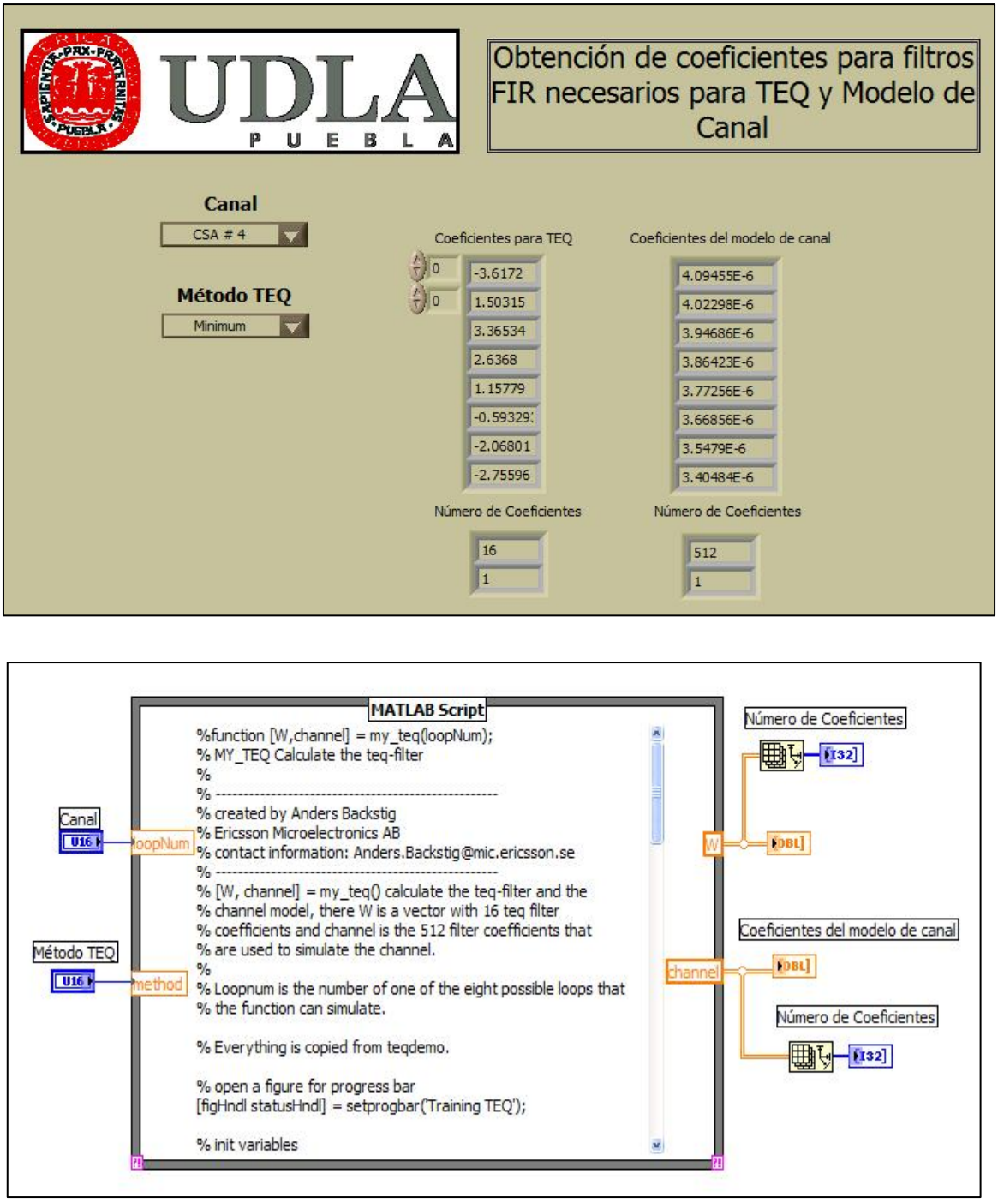

Figura 4.10 Instrumento virtual para la obtención de los coeficientes de los filtros FIR.

4.3.1 Ruido en el canal

En el capítulo 2 se explicaron las diferentes fuentes de ruido que afectan el canal de transmisión en un sistema ADSL. El proceso de modelar el ruido resulta complejo, por lo tanto, se tomó un modelo de ruido existente en la literatura [BAC01] y desarrollado en MATLAB. La función *add_noise* del *toolbox* de ADSL para MATLAB toma el modelo de ruido y genera un vector ruido que se suma al vector símbolo DMT que contiene 544 muestras. El tipo de ruido que se suma es del tipo blanco gaussiano (AWGN – *Additive White Gaussian Noise*), que comúnmente se utiliza para modelar el ruido proveniente de una combinación de varias fuentes de ruido.

La Figura 4.11 muestra la programación de este modelo en LabVIEW, donde el *script* del lado izquierdo es el modelo de ruido y el *script* de la derecha es la función que adhiere ruido al símbolo DMT. Los algoritmos completos de esta programación se muestran en el Apéndice.

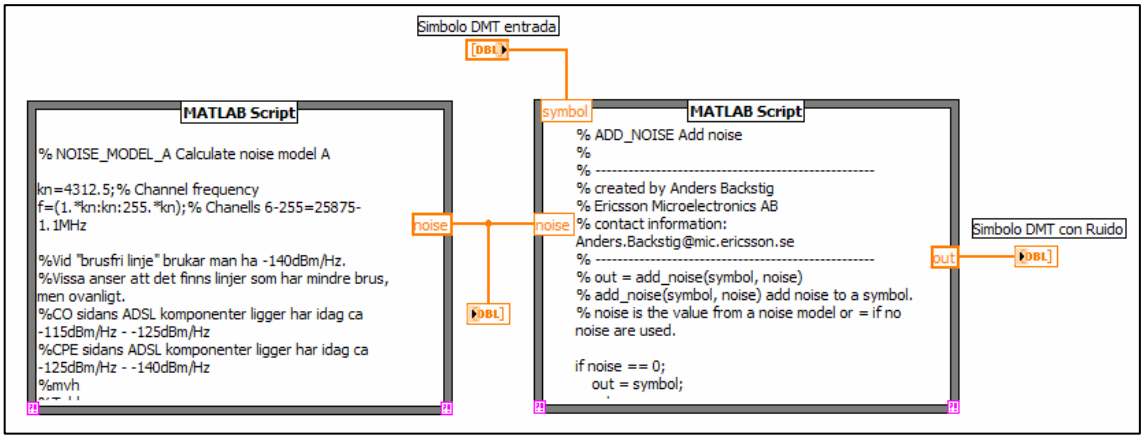

Figura 4.11 Instrumento virtual para adición de ruido AWNGN al símbolo

4.4 SIMULACIÓN DEL RECEPTOR DMT

El bloque más importante en el receptor es el ecualizador en el dominio del tiempo, cuya tarea principal es reducir la duración de la respuesta al impulso del canal de

transmisión a los niveles de duración del prefijo cíclico. Este ecualizador se modelo con un filtro FIR de 16 *taps* cuyos coeficientes se obtienen del *toolbox DMTTEQ* de MATLAB.

Existen diversos métodos de diseño para el ecualizador en el dominio del tiempo (Métodos TEQ), sin embargo, la elección de un método en particular depende de lo que se prefiera optimizar. Estos métodos se clasifican en dos grandes enfoques: por un lado, aquellos métodos que minimizan el error cuadrado medio (*mean squared error*) y por otro lado, los métodos que minimizan la energía fuera de la respuesta cortada del canal (*Shortened Channel Response*) [ARS00].

El *toolbox DMTTEQ* es una colección de funciones desarrolladas en MATLAB para diseñar y probar varios métodos de diseño para un ecualizador en el dominio del tiempo. En esta simulación de ADSL se utilizan dos métodos diferentes de los múltiples disponibles. Estos métodos son los siguientes:

- Mínimo error cuadrado medio – Limitador de energía
  (*Minimum mean squared error – unit energy constraint MMSE_UEC)*

- Mínima interferencia entre símbolos
  (*Minimum Intersymbol Interferente – MISI)*

El instrumento virtual de la Figura 4.10 obtiene los coeficientes para cada uno de estos métodos de diseño. Por cada uno de ellos se obtienen los coeficientes correspondientes a cada uno de los canales estándares (*CSA Loops*). Es así como se obtienen tanto los 512 coeficientes para un canal determinado como los 16 coeficientes para el ecualizador diseñado con un método determinado.

El programa principal de esta simulación acepta como variables de entrada el modelo de canal (*CSA Loops 1-8)* y el método deseado de ecualización (Método TEQ), para que el usuario tenga las opciones de simular la transmisión de datos sobre diferentes

modelos de canal, así como la oportunidad de elegir entre tres métodos de ecualización diferentes.

La simulación del canal de transmisión se desarrolla con una operación de convolución entre los coeficientes del canal y la secuencia de símbolos que se transmite por el canal. Para ello se utiliza la función *filter* de MATLAB que sirve para filtrar la señal de entrada a través de un filtro FIR definido por los coeficientes del canal. La Figura 4.12 muestra el código fuente de este instrumento virtual.

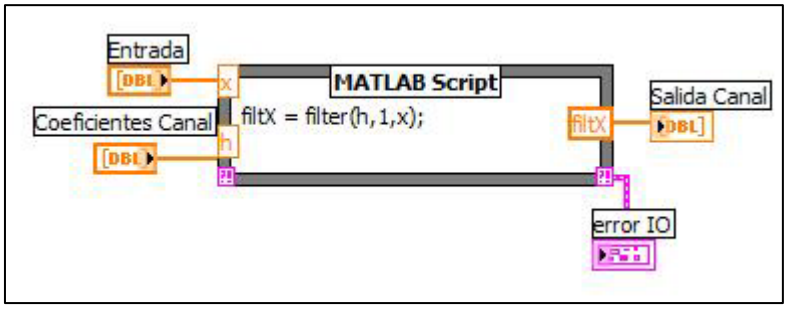

Figura 4.12 Instrumento virtual del filtro FIR para modelar el canal

El ecualizador en el dominio del tiempo también se modela como un filtro FIR, utilizando la misma función *filter*. El código fuente se muestra en la Figura 4.13

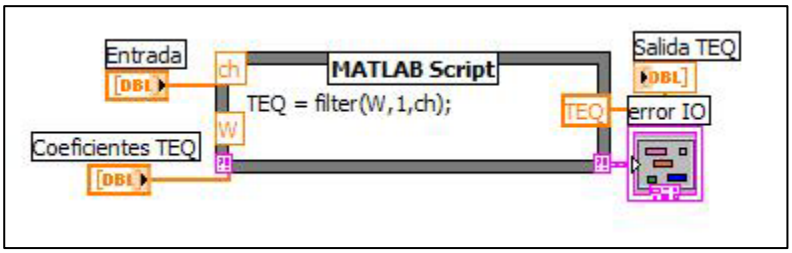

Figura 4.13 Instrumento virtual del filtro FIR para TEQ

Los bloques restantes del receptor realizan el proceso inverso al descrito en el transmisor, por lo tanto los algoritmos son básicamente los procesos inversos. Las Figuras 4.14, 4.15 y 4.16 muestran los instrumentos virtuales para el eliminador de prefijo cíclico, la FFT y el eliminador de espejo de datos, respectivamente.

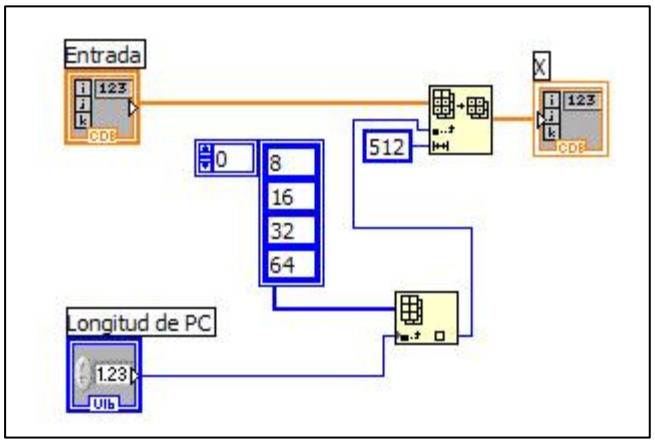

Figura 4.14 Instrumento virtual de eliminador de prefijo cíclico.

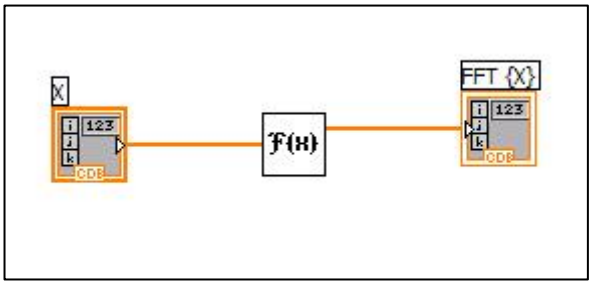

Figura 4.15 Instrumento virtual de FFT.

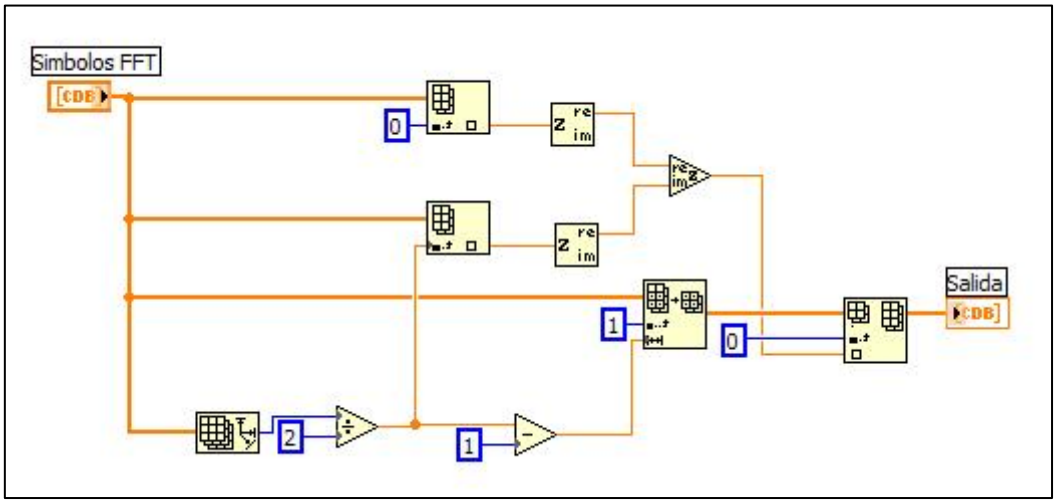

Figura 4.16 Instrumento virtual de eliminador de espejo de datos.

El banco de decodificadores QAM se desarrolló con la función *qamdemod* de MATLAB y de manera similar al banco de codificadores QAM, las iteraciones de un lazo *for* son equivalentes a los 256 decodificadores. La Figura 4.17 muestra la programación de este bloque de salida.

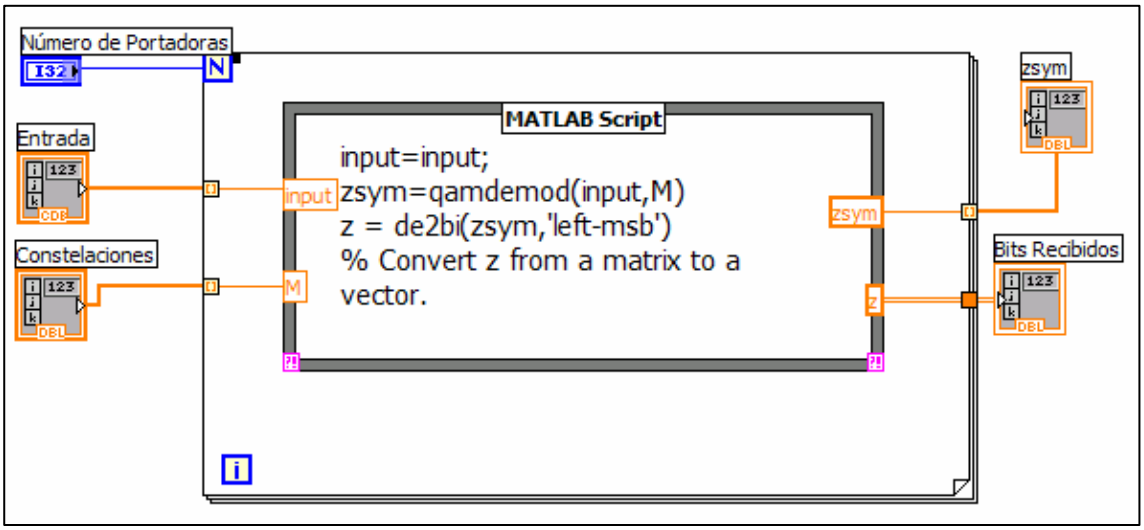

Figura 4.17 Instrumento virtual del banco de decodificadores QAM

Para calcular el BER, se desarrollaron dos instrumentos virtuales. Uno de ellos calcula el BER para cada uno de los simbolos transmitidos, en tanto que el otro calcula el BER global, es decir, el BER de todos los símbolos transmitidos. Para la realización de estos dos instrumentos virtuales se hizo uso del operador XOR, cuya tabla de verdad se muestra en la Tabla 5.1:

Tabla5.1 Tabla de verdad XOR

X	Y	Salida X*Y
0	0	0
0	1	1
1	0	1
1	1	0

Este operador regresa 1 si las entradas X y Y son diferentes, por lo tanto es una operación adecuada para realizar comparaciones entre dos números. En este caso, los instrumentos virtuales realizan la comparación de los dos vectores que contienen los bits transmitidos y los bits recibidos y almacenan el resultados en un tercer vector. Para contabilizar el número de errores detectados en el receptor basta con sumar los unos contenidos en el vector resultado de la operación XOR. Para calcular BER se divide el número de errores detectados entre el número total de bist transmitidos. Para el caso del instrumento que calcula el BER global, el número de errores se calcula fuera del lazo principal de ejecución, en tanto que para el cálculo del BER por cada simbolo DMT éste se calcula durante cada una de las iteraciones del lazo. La Figura 4.18 muestra el diagrama a bloques del instrumento que calcula el BER global, sin embargo es básicamenmte el mismo algoritmo para el BER por cada simbolo DMT.

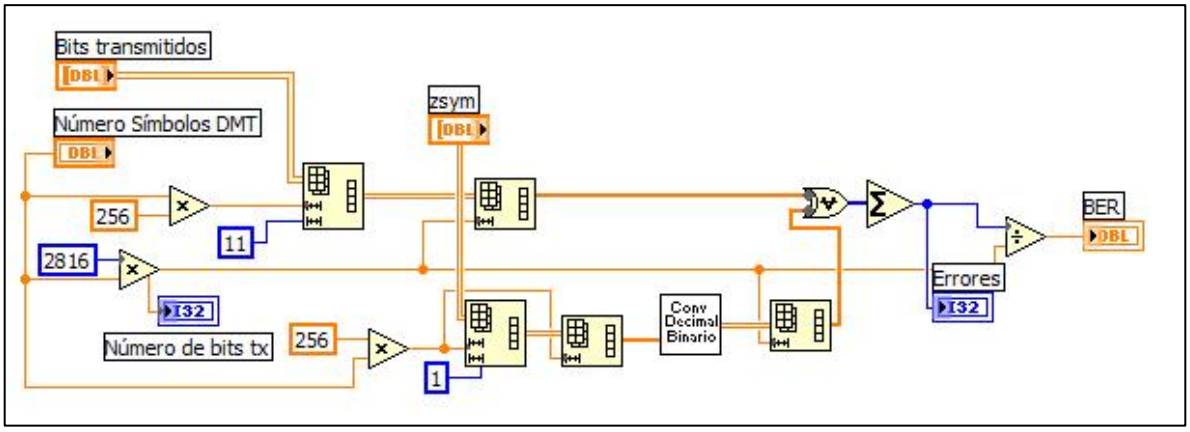

Figura 4.18 Instrumento virtual para calcular el BER.