Arquitectura estimador de canal y ecualizador

Cómo plantear la arquitectura de los bloques para su descripción hardware

Hipólito Guzmán Miranda
Departamento de Ingeniería Electrónica
Universidad de Sevilla
hguzman@us.es

Resumen

Se describe a continuación la arquitectura del estimador de canal que se debe realizar y se proponen una serie de mejoras sobre la misma. Posteriormente se describe una posible arquitectura para el ecualizador, junto con una serie de consideraciones que se deben realizar al diseñarlo. Finalmente, se propone una arquitectura para realizar los testbenches de los módulos completos que nos permita realizar comparaciones con respecto a los resultados obtenidos en Matlab.

Estimador de canal

El estimador de canal utiliza el interpolador lineal desarrollado anteriormente para realizar la estimación del canal en las posiciones existentes entre dos pilotos. Para cada pareja de pilotos, el estimador debe realizar las siguientes acciones:

1.- Determinar si el piloto inferior transmitido era positivo (+4/3) o negativo (-4/3). Esta información la debe extraer del bloque PRBS (Pseudo-Random Binary Sequence)
2.- Determinar si el piloto superior transmitido era positivo (+4/3) o negativo (-4/3), utilizando también el bloque PRBS.
3.- Determinar cuál ha sido el efecto del canal en el piloto superior. Ya que, en el dominio de la frecuencia, Y=X*H, podemos despejar: H = Y/X, Por lo que Hinf = Yinf /

Xinf[1]. Debido a que un factor de escala de 3/4 se puede compensar más adelante[2], una simplificación muy recomendable es simplemente ajustar el valor en función del signo del piloto enviado, es decir, calcular $H_{escalado}$ = +/- Y.

4.- Realizar el paso 3 para el piloto inferior. Aplicando la simplificación anteriormente mencionada, se trata de un ajuste de signo en función de lo que indique el PRBS. Si el piloto enviado fue positivo, se mantiene el signo; y si el piloto enviado fue negativo, se invierte el signo.

5.- Una vez que se tienen Hinf y Hsup, pasar estos valores al interpolador e iniciar el proceso de interpolación. La salida del interpolador contendrá los valores del canal estimado entre ambos pilotos.

Una posible arquitectura para el estimador de canal es la siguiente:

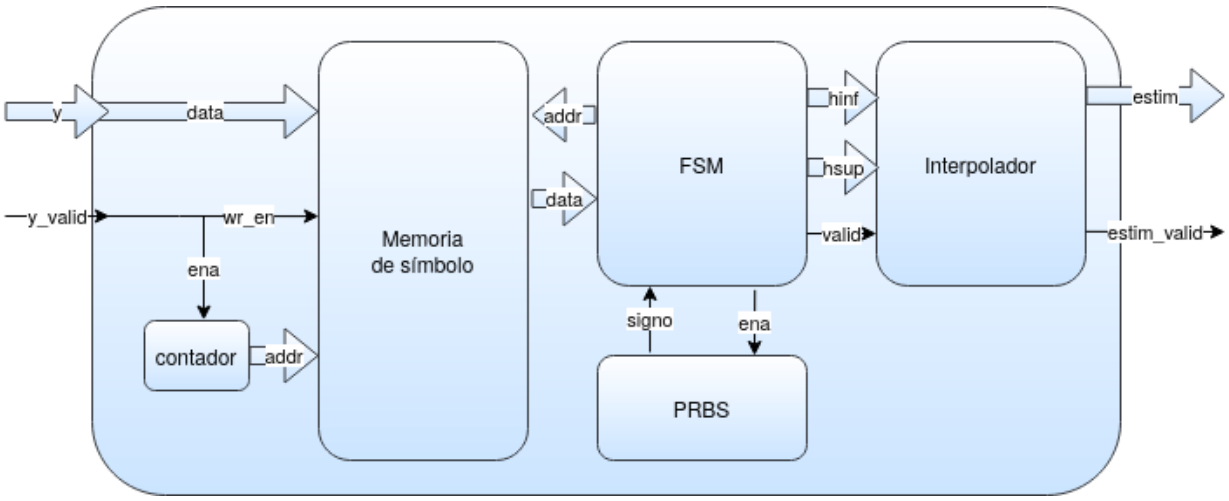

Fig. 1: Arquitectura propuesta para el estimador de canal

La máquina de estados (FSM) debe leer los pilotos de la memoria y desplazar el PRBS cuando sea necesario.

[1] Nótese la conveniencia de que los pilotos tomen valores reales (+4/3 y -4/3), ya que de esa forma no es necesario realizar una división por un número complejo.

[2] Recordemos también que nuestra nuestra implementación del interpolador ya incluye un factor de escala de 12/16.

Un posible diagrama de estados (simplificado) de la máquina de estados puede ser:

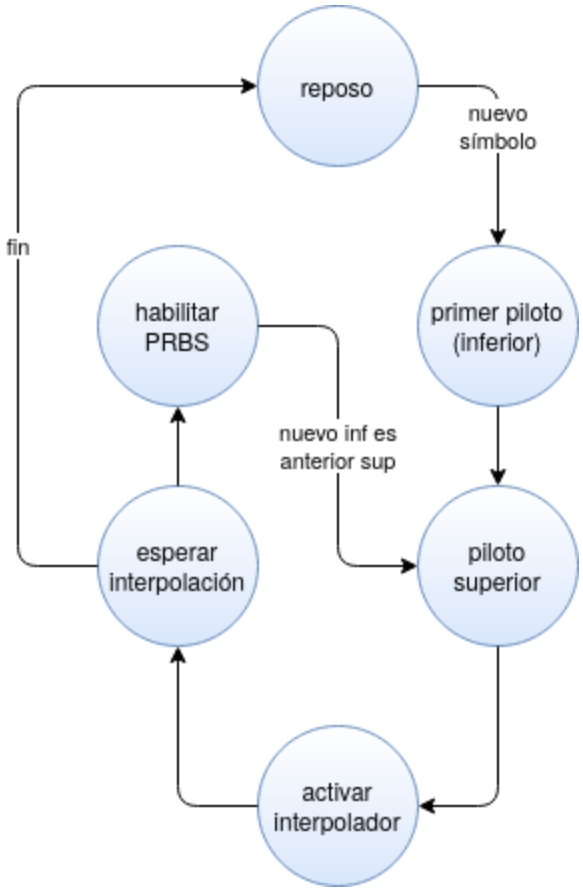

Fig 2. Diagrama de estados (simplificado) del estimador de canal

Se pueden optimizar los tiempos si habilitamos el PRBS a la vez que está funcionando el interpolador (con cuidado de no cambiar las entradas del interpolador mientras este esté funcionando). Además, como el piloto superior de un grupo de 12 portadoras será el piloto inferior del siguiente grupo, no será necesario ajustar su signo dos veces, por lo que a partir del segundo grupo de portadoras, sólo necesitamos ajustar el signo del piloto superior.

Se deja como parte del trabajo del alumno determinar qué otras señales o conexiones pueden ser necesarias, por ejemplo la FSM no debe empezar a leer datos de la memoria hasta que se escriba todo el símbolo completo, o al menos hasta que no estén escritas las primeras 12 portadoras.

Una simplificación muy interesante es sustituir la memoria en la que almacenamos un símbolo entero por una memoria con sólo 12 posiciones. De esta forma la arquitectura, gestionando bien cuándo escribimos en dicha memoria, será válida tanto para símbolos de 2k como para símbolos de 8k portadoras, ocupando muchos menos recursos hardware.

Se proporcionan bloques de memoria descritos en VHDL (al sintetizarlos, el sintetizador inferirá una memoria de bloque) aquí: http://woden.us.es/docs/docencia/EDC1MIT/edc.tar.gz (al descomprimir, carpeta 'bram'). Se proporcionan memorias de un puerto y de doble puerto. (También se puede utilizar el "block memory generator" proporcionado por el Xilinx IP Core Generator, aunque si queremos simularlo con GHDL deberemos compilar las librerías unisim con dicho simulador.)

Ecualizador

El ecualizador se encarga de aplicar a las portadoras recibidas (Y) la operación inversa a la realizada por el canal (1/H) con el objetivo de obtener las portadoras enviadas (H). La precisión del resultado de esta operación dependerá de la precisión con la que se haya estimado el canal.

Para poder realizar la división compleja que nos permitirá calcular 1/H, hay que considerar que para realizar la división compleja se debe multiplicar tanto el numerador como el denominador de la fracción por el conjugado de H.

Dicho de otra forma, multiplicamos por 1, expresando ese 1 como una división de dos números iguales (y distintos de cero). Elegimos como número el que nos asegura que nos quedará en el denominador un número entero. Ese número es el conjugado de H:

$$\frac{1}{H} = \frac{1}{a+bi} = \frac{1}{a+bi} \times \frac{a-bi}{a-bi} = \frac{a-bi}{a^2+abi-abi-b^2i^2} = \frac{a-bi}{a^2+b^2} = \frac{conj(H)}{|H|^2}$$

De esta forma, podemos plantear una arquitectura para el ecualizador utilizando multiplicaciones y divisores.

Una posible arquitectura puede ser la siguiente:

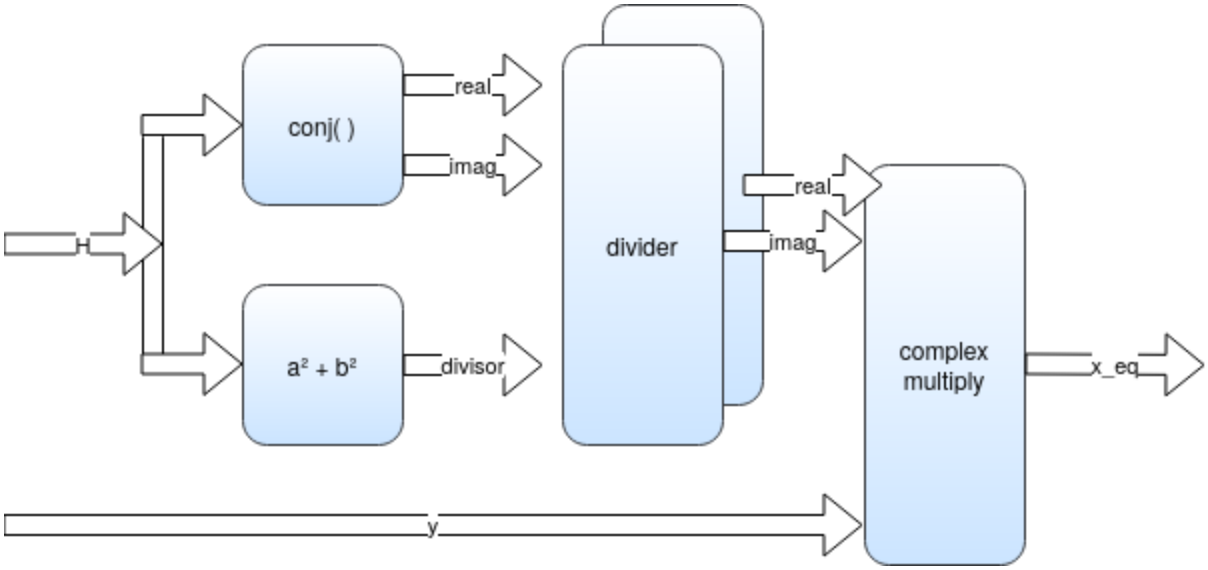

Fig 3. Procesado de señal en el ecualizador

Se deja como parte del trabajo que tiene que realizar el alumno el determinar cómo controlar el flujo de datos por el sistema. El ecualizador tiene que leer de dos memorias de bloque: la que contiene el símbolo recibido y la que contiene el canal estimado, y debe esperar a que el divisor termine de operar antes de alimentarlo con más datos. Debido a que la memoria que contiene el símbolo recibido ya tendrá sus dos puertos utilizados, se debe multiplexar uno de los dos puertos, por ejemplo, eligiendo si la dirección que va al puerto B de la memoria es la que viene del estimador o la que viene del ecualizador en función de cuál de los dos esté trabajando en ese momento.

Si el numerador es más pequeño que el denominador, será necesario insertar un factor de escala para poder realizar la división. El factor de escala más sencillo de insertar es una multiplicación por 2^N, es decir, un desplazamiento.

Se proporciona un bloque divisor desarrollado en VHDL aquí: http://woden.us.es/docs/docencia/EDC1MIT/edc.tar.gz (al descomprimir, carpeta 'divider'). (También se puede utilizar el "divider generator" proporcionado por el Xilinx IP Core Generator, aunque si queremos simularlo con GHDL deberemos compilar las librerías unisim con dicho simulador.)

Verificación de los bloques

Tanto el estimador de canal como el ecualizador deben verificarse introduciéndoles como entrada al menos un símbolo OFDM completo, y comprobando que la salida de ambos módulos coincide en la medida de lo esperado con las salidas de Matlab.

Se propone la siguiente arquitectura de testbench utilizando el paquete vhdl-verification, disponible en la página de la asignatura:

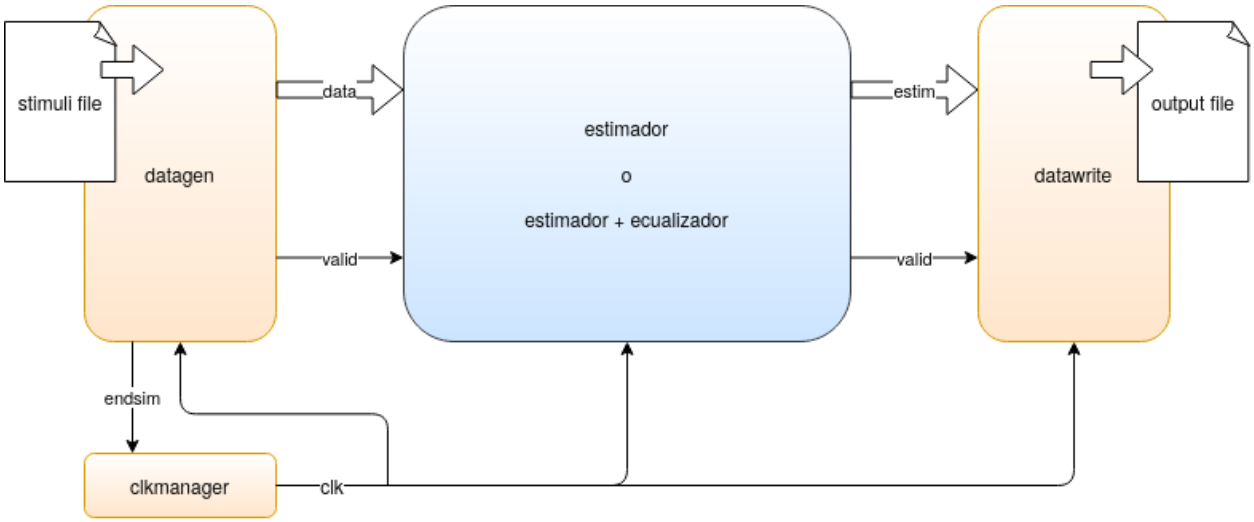

Fig 4. Arquitectura de testbench utilizando el paquete vhdl-verification.

Se recomienda encarecidamente la lectura de la documentación del package[3], en la que vienen explicados los tipos de datos y función de todos los generic y port de las distintas entidades.

Para mejorar la confianza en el procesado de señal realizado por el diseño, se recomienda encarecidamente no sólo que se dibujen las gráficas resultantes, sino también que se calculen los errores absolutos y relativos del procesado en VHDL con respecto al procesado en Matlab.

Se deja como ampliación opcional para subir nota el uso de las capacidades de verificación explicadas en clase para mejora de los testbenches o desarrollo de los testbenches de los bloques constituyentes como el PRBS, utilizando OSVVM/UVVM u otros.

[3] Documentación disponible en: http://woden.us.es/docs/vhdl-verification/. Asimismo, se pueden ver algunos ejemplos de instanciación de las entidades del package en: http://woden.us.es/docs/vhdl-verification-files/test/