Filtrado de una señal analógica mediante VHDL

Luca Scalambrin
Laboratorio III - Electrónica digital
Ing. en Telecomunicaciones - Instituto Balseiro, UNCuyo, CNEA, Argentina

6 de octubre de 2019

En el presente trabajo se llevó a cabo el filtrado de una señal de voz mediante lenguaje VHDL y una placa de desarrollo de FPGA Nexys 3 - Spartan 6. Para ello se debió muestrear la señal de interés a 20 KHz, para luego convolucionar las muestras con los coeficientes de un filtro pasa bajos FIR. Para transmitir los datos hacia la computadora se dio uso de la UART de la placa, para luego analizarlos mediante el software *audacity*.

1. Especificación

1.1. UART

Implementar un módulo para enviar datos a 20 KHz, con una baud rate de 460,000, con 8 bits de datos y 1 bit de parada. Testear el módulo utilizando un contador.

1.2. PmodMic

Implementar un módulo para samplear micrófono a 20 KHz, siendo los 8 bits más significativos los portadores de la información. Con otro módulo leer los datos generados por el módulo PmodMic y enviarlos a la PC mediante la UART. Una vez obtenidos los datos en la PC, analizarlos con algún software de audio. El software utilizado fue *audacity*.

1.3. Filtro

Diseñar un filtro tipo FIR pasabajos, convirtiendo sus coeficientes a punto fijo en binario. Luego, mediante la utilización de arrays y genéricos se deberá realizar la convulción de las muestras tomadas por el micrófono y los coeficientes del filtro.

2. Diseño del módulo top

En la Fig. 1 se observa el diagrama del módulo TOP diseñado. Este cuenta con seis módulos, los cuales corresponden a cada una de las secciones que se necesitan para llevar a cabo el filtrado de la señal.

Las entradas del TOP son tres, el clock, el bit de reset y la lectura de voz empleada por el micrófono.

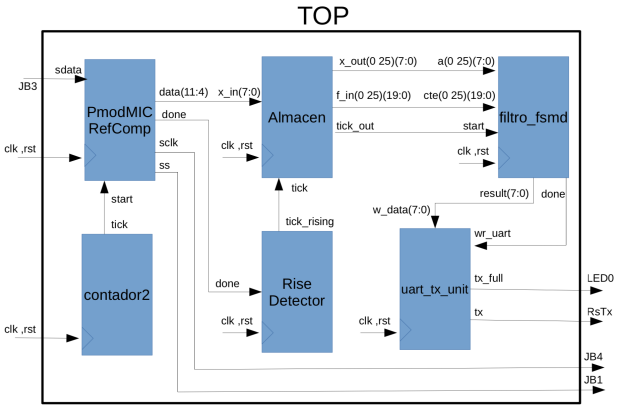

Figura 1: Diagrama en bloques del móludo top del sistema empleado. Se observan los seis módulos empleados con sus correspondientes entradas y salidas.

Las salidas del módulo son cuatro, tx y tx_full, las cuales corresponden a los datos en serie y a la señal de error correspondiente; y las señales de entrada del micrófono físico $sclk$ y ss.

3. Diseño, implementación y testing de cada módulo

A continuación se detallan cada uno de los módulos internos del módulo top. Tanto el código VHDL como el testbench correspondiente a los módulos se adjuntan con el presente informe. Además, en el repositorio digital [1] se podrán encontrar todos los códigos correspondientes.

3.1. Módulo PmodMICRefComp

El micrófono empleado se comunica con el TOP mediante interface SPI, pero para filtrar la señal se necesitan los datos en paralelo, por lo que el módulo PmodMICRefComp será el encargado de realizar esta función. En la Fig. 2 se encuentra la máquina de estados empleada en el módulo, la cual se deberá iniciar cada 5000 ciclos de reloj, de modo de obtener un muestreo del micrófono de 20 KHz.

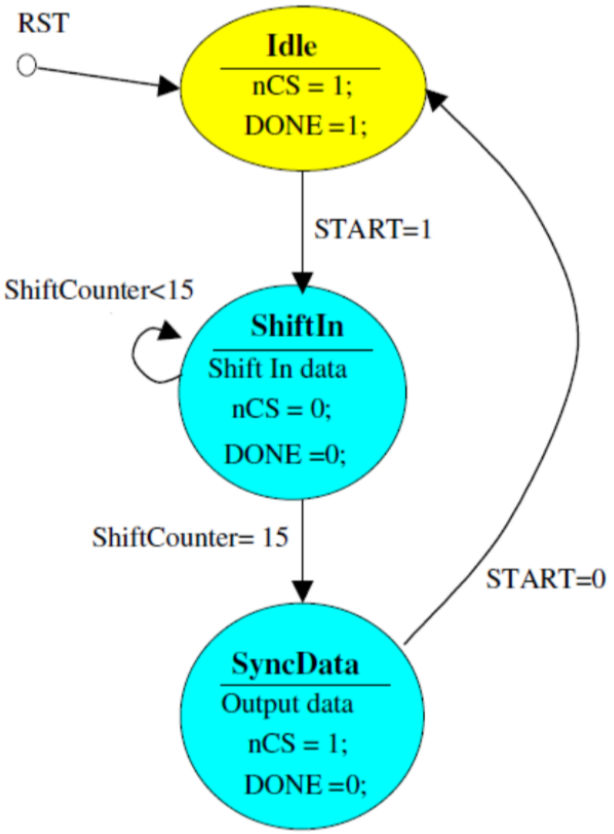

Figura 2: Máquina de estados (FSM) empleada en el módulo $PmodMICRefComp$.

Una vez obtenida la palabra de 12 bits, se encontrará disponible en la salida $data(11 : 0)$, y *done* cambiará de "0" a "1".

El código de este módulo fue provisto por la cátedra, y se podrá consultar en el respositorio digital [1] como $PmodMICRefComp$.

3.2. Módulo contador2

El módulo *contador*2 es el encargado de generar una señal *tick* cada 5000 ciclos del clock de la placa, de modo de obtener muestras cada 20 KHz. La particularidad es que la señal *tick* tendrá el valor de "1" durante 8 clocks, siendo esto lo necesario por el módulo $PmodMic$.

Para el testing del módulo se corroboró que cada 5000 clocks, la señal *tick* tome el valor "1" durante 8 clocks, y que luego tome el valor "0" por 4992 clocks. El código del testeo posee el nombre *tb_cuonter* y el del módulo es *contador*2, los cuales se pueden consultar en el respositorio digital [1].

3.3. Módulo Almacen

El módulo Almacen es el encargado de generar los dos vectores a convolucionar para poder filtrar la señal, y la señal de *start* para el filtro. La lógica empleada es similar a la de un "Shift Register", que a partir de las muestras de 8 bits que ingresen al módulo, generará un vector $x\_out(0\ \ 25)(7 : 0)$ con 26 de estas muestras, las cuales se irán "shifteando" cada vez que ingrese una nueva.

Los valores de $f\_in(0\ \ 25)(19 : 0)$ son los correspondientes a las 26 constantes del filtro empleado y la señal *tick_out* será la señal de *start* del filtro, la cual se generará una vez que el vector x_out se encuentre listo para ser convolucionado.

Cabe aclarar que, para poder utilizar los arreglos, fue necesario la utilización del paquete provisto por la cátedra, llamado "$filtro\_paq.vhd$"; en el cual se declaran los arreglos empleados.

Para el testing se colocaron palabras de 8 bits en la entrada, y se corroboró que en cuanto $tick = 1$, estas se vayan incorporando al vector x_out. El código del testeo posee el nombre *tb_almacen* y el del módulo es *almacen*, los cuales se pueden consultar en el respositorio digital [1].

3.4. Módulo Rise Detector

El módulo Rise Detector se encarga de generar una señal *tick_rising* que tendrá el valor de "1" cuando el módulo $PmodMICRefComp$ haya generado la muestra.

Para el testing del módulo se alteró el valor de la señal *done* de "0" a "1" y luego se corroboró que se haya detectado dicho cambio mediante la señal *tick_rising*. El código del testeo posee el nombre *tb_rising* y el del módulo es *rise_detector*, los cuales se pueden consultar en el respositorio digital [1].

3.5. Módulo Filtro

El módulo filtro es el encargado de, una vez que la señal *start* se activa, efectuar la convolución de las dos entradas $a(0\ \ 25)(7:0)$ y $cte(0\ \ 25)(19:0)$, para luego generar la señal de *done* y el resultado del filtrado en la señal $result(7:0)$.

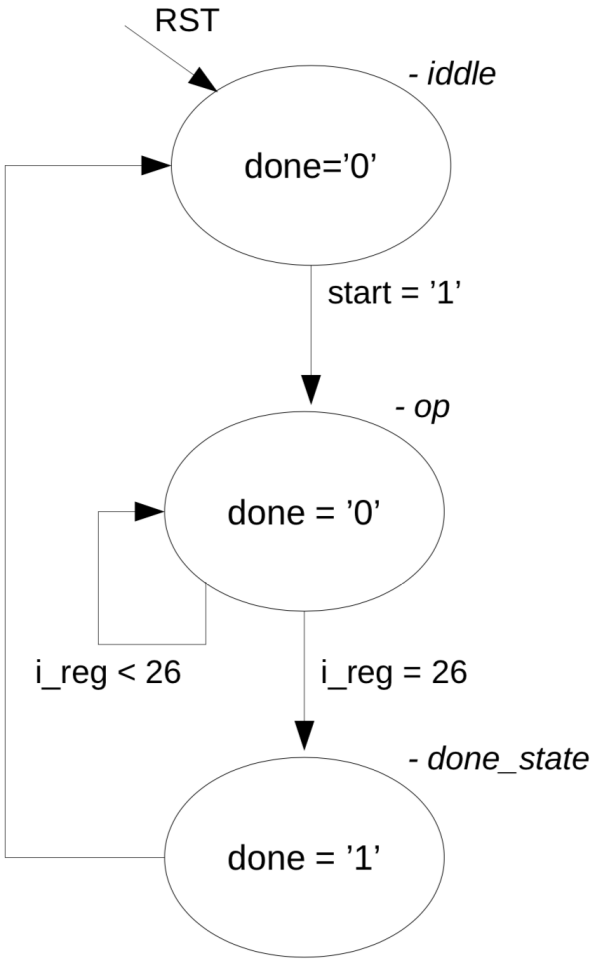

Figura 3: Máquina de estados (FSM) empleada en el módulo *Filtro*.

Para este módulo se diseñó la máquina de estados que se observa en la Fig. 3. Cuando el módulo *Almacen* genera la señal *start*, el filtro acumula las multiplicaciones que se efectúan entre los elementos de igual índice de los dos vectores, $a(0\ \ 25)(7:0)$ y $cte(0\ \ 25)(19:0)$. Una vez acumuladas las 26 multiplicaciones, se genera la señal *result* con los 8 bits correspondientes a las posiciones (27:20).

Para el testing se cargaron los valores correspondientes a un filtro de orden 9 y deltas en cada una de las muestras, de modo comparar el valor esperado con el de la señal de salida *result*. El código del testeo posee el nombre tb_filtro y el del módulo es $filtro\_fsmd$, los cuales se pueden consultar en el respositorio digital [1].

3.6. Módulo UART

El módulo UART se encarga de, una vez obtenida la muestra filtrada, transmitirla en serie mediante la salida tx. Para ello es necesaria una señal wr_uart que será generada por el filtro.

Al igual que con el módulo *PmodMICRefComp*, este módulo fue provisto por la cátedra y lo único que se debió realizar fue la configuración. El código se puede encontrar con el nombre "$uart\_tx\_unit$"en el respositorio digital [1].

4. Test de integración total

Una vez completos todos los módulos, se instanciaron en el top, de modo de poder testear el sistema de filtrado por completo. En la Fig.4 se pueden observar los mapeos llevados a cabo, estos son, la señal de error tx_full con el LED00, la señal a transmitir tx con RsTx de la UART y las señales ss, $sdata$ y $sclk$ con los correspondientes pines de los conectores JB. Todos estos mapeos se encuentran realizados en el archivo "Nexys3_Master.ucf".

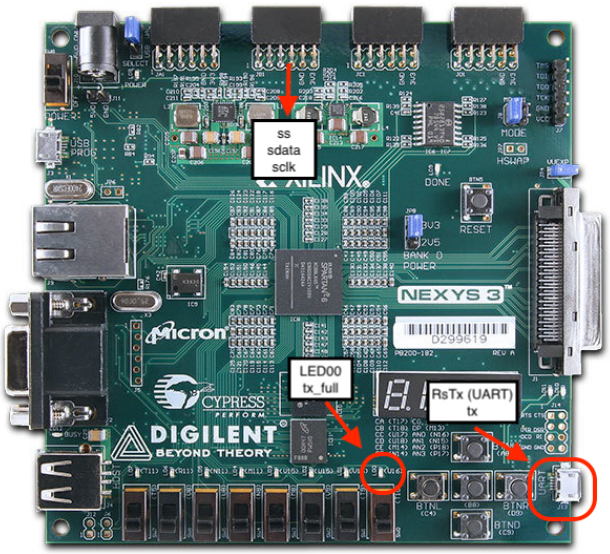

Figura 4: Esquema de la placa de desarrollo utilizada y los respectivos mapeos realizados.

Lo que se realizó para testear el sistema por

completo fue generar un tono de 5 KHz y sumarlo a la voz humana. De esta forma, al filtrar dicho audio, se comprobó que los tonos de mayores frecuencias se hayan atenuado. Esto efectivamente ocurrió, y el análisis del espectro se puede observar en la Fig. 5, donde se observa que el tono en la frecuencia de 5 KHz esta atenuado en 20 dB respecto a los tonos en frecuencias menores a los 3,5 kHz.

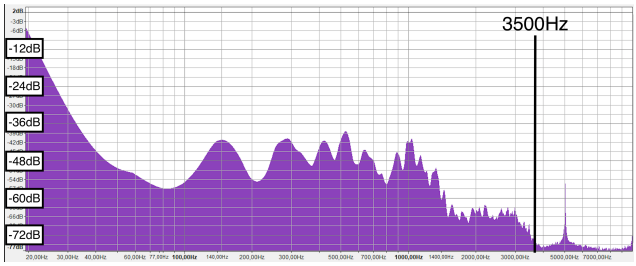

Figura 5: Espectro obtenido luego de haber realizado el filtrado de la señal adquirida mediante el micrófono.

5. Conclusiones

Mediante la utilización del lenguaje $VHDL$ y conceptos de diseño y programación modular se pudo implementar el sistema de filtrado que cumple con la especificación. Para ello fue muy útil la utilización de módulos ya realizados, como el del contador. Además, mediante la metodología de diseño sincrónico y la implementación de una máquina de estados (FSM) con datapath fue posible llevar a cabo la convolución de las muestras del micrófono y los coeficientes del filtro

Las pruebas fueron realizadas en una placa de desarrollo de FPGA Nexys 3 - Spartan 6, con el micrófono que provee la misma compañía que diseña la placa.

Todos los códigos desarrollados se pueden consultar en el repositorio digital de GitHub titulado "FiltroSignalAnalogica" [1].

Referencias

[1] Scalambrin Luca. *"Laboratorio III - Filtro señal analógica"*. https://github.com/LucaScalam/FiltroSignalAnalogica.git