COMISIÓN NACIONAL
DE ENERGÍA ATÓMICA

UNIVERSIDAD
NACIONAL DE CUYO

Trabajo Práctico – Módulo de Procesamiento para Localización de Fuente Sonora

Introducción a la Microfabricación y FPGA
Instituto Balseiro
6 de Diciembre, 2013

Ing. Ramiro García

Introducción

El objetivo de este trabajo es diseñar e implementar un módulo en lenguaje VHDL que reciba las señales de dos micrófonos espacialmente separados y determine la diferencia temporal entre ambos, permitiendo enviar esa información a otro modulo para el cálculo de la dirección de la fuente sonora. Al usar solo dos micrófonos la posible ubicación de la fuente sonora para una determinada diferencia de tiempos queda determinada por hipérbolas [1], como se aprecia en la figura 1. Es posible agregar uno o más micrófonos [1] y mediante la intersección de las hipérbolas determinar la ubicación de la fuente (figura 2).

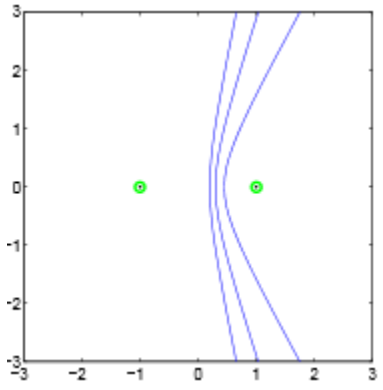


Figura 1 - Hipérbolas de dt constante para una configuración de dos micrófonos

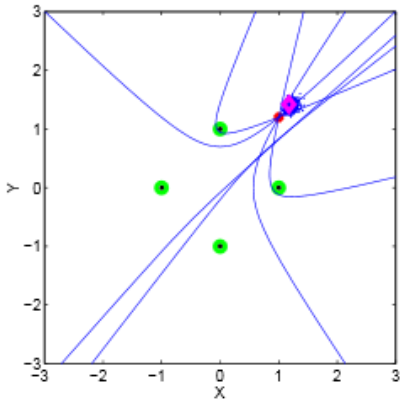


Figura 2 – Array de micrófonos

El módulo de procesamiento de este trabajo forma parte de un sistema global, conformado por 4 módulos en total. Un módulo de adquisición de las señales de los micrófonos, uno de procesamiento, uno de control de la FIFO para la comunicación con la PC y por ultimo uno que muestre en una pantalla lcd la información del módulo de procesamiento (figura 3).

El mismo recibe como entradas las señales done1 y done2 de los micrófonos 1 y 2 respectivamente, que habilita a los detectores (proceso sincrónico con la frecuencia de muestreo) a leer los datos de los micrófonos data1 y data2.
Como salidas se necesitan una señal *enable* que avise cuando se identifica el ruido de la fuente sonora (aplauso) en el primero de los micrófonos, una señal que envíe a la pantalla cuál de los micrófonos se activó primero (*Tickfirst*), una que envíe al módulo de la pantalla el tiempo que transcurrió entre detecciones (*dt* de 8bits) y por último una señal que avise que se terminó de medir el tiempo entre detecciones (*done*).

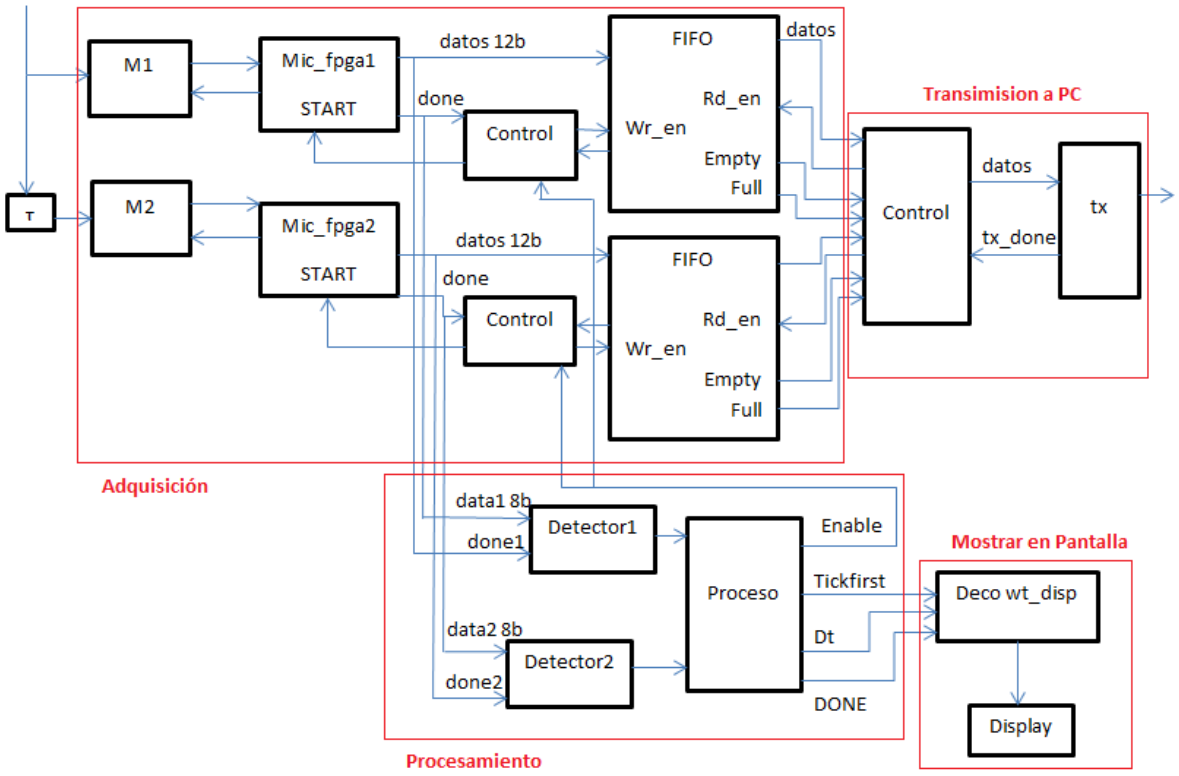


Figura 3 – Diagrama de bloques del sistema completo

Desarrollo del trabajo

Módulos de detección (detector1 y detector2)

En un principio se pensó utilizar la medida de la varianza del ruido de fondo como referencia para detectar la fuente sonora de interés, se estudió esta posibilidad utilizando una grabación de un aplauso (figura 4). El algoritmo se programó en el lenguaje de programación python, por su rapidez a la hora de implementar:

```
import numpy as np
f=open("aplauso.txt","rb")
data=f.read()
a=np.array([int()])
a2=np.array([int()])
for i in range(512):
        signal=int(data[i*2:(i+1)*2],16)
        a=np.append(a, signal)
        a2=np.append(a2,signal*signal)
#print i,a
i=i+1
prom=np.average(a)
```

```
prom2=np.average(a2)
var=prom2-prom*prom
signal=int(data[i*2:(i+1)*2],16)
while abs((signal-prom)*(signal-prom)) < abs(100*var):
        i=i+1
        signal=int(data[i*2:(i+1)*2],16)
        a=np.append(a[1:], signal)
        a2=np.append(a2[1:],signal*signal)
        prom=np.average(a)
        prom2=np.average(a2)
        var=prom2-prom*prom
print i,prom, prom2,var,(signal-prom)*(signal-prom)
```

El algoritmo toma 512 valores de señales de la grabación y calcula la varianza móvil de la muestra. El número de valores resulta de que tomando una muestra más chica la varianza puede dar cero, al mantenerse constante la señal. Una vez calculada la varianza se compara el último valor de la señal con la varianza móvil y de ser X veces más grande, se deduce que es un ruido que no es de fondo.

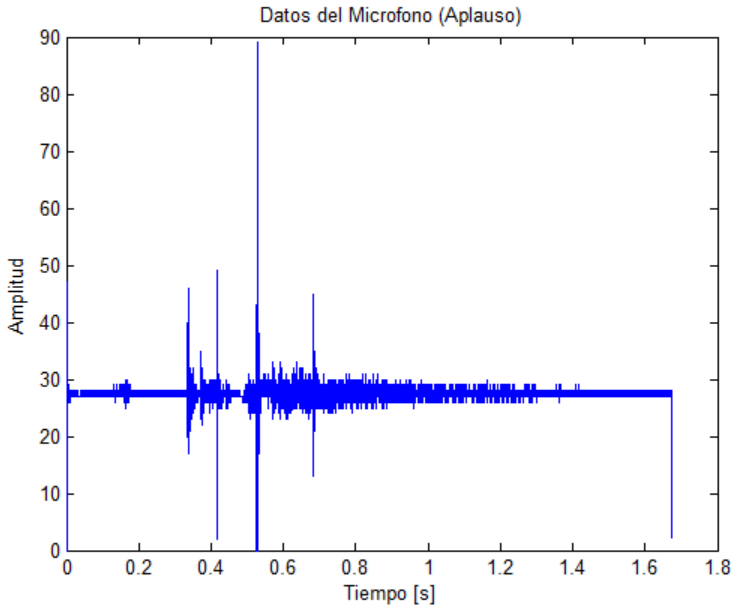
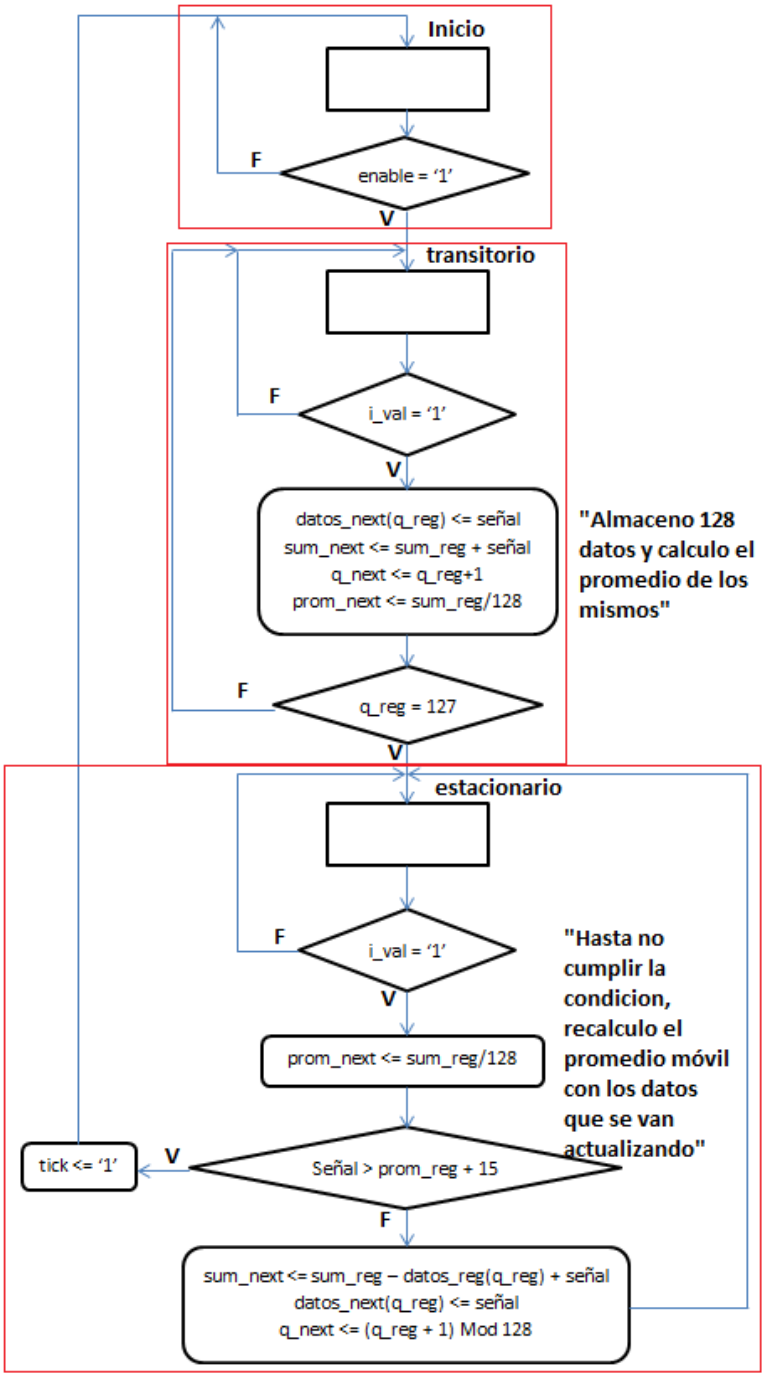


Figura 4- Datos de amplitud de aplauso (aprox. a los 0.3 segundos)

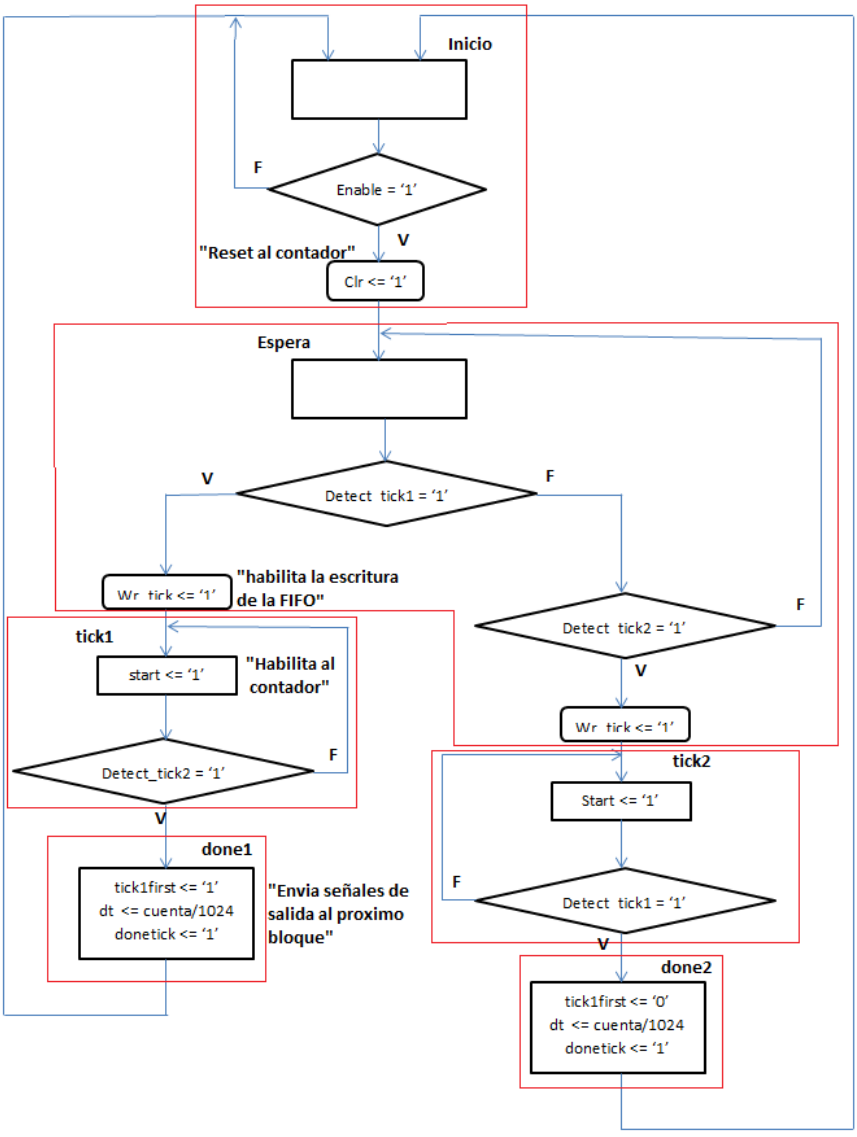
Como el cálculo de la señal implica almacenar 512 valores y conlleva varias multiplicaciones en cada iteración, se decidió directamente comparar la señal con el promedio de la misma. Debido a la forma de la señal (no es tan ruidoso el ambiente) simplemente definiendo un threshold por encima del promedio de la señal uno puede determinar cuándo un ruido que no es de fondo es captado por el micrófono. El threshold

en un principio se fijara 15 unidades por encima del promedio móvil, esto se definió basándonos en la observación de la figura 4. El promedio aproximadamente 28 y el threshold quedaría definido por lo tanto en 43, el ruido externo en t = 0.3s es mayor a 43 por lo que será detectado. A continuación se muestra un diagrama ASM del circuito:



<u>Módulo de medición de la diferencia temporal entre micrófonos (Proceso)</u>

Este módulo recibe como entrada los ticks de los detectores 1 y 2, y tiene como salidas las mismas que las del módulo de procesamiento. Estas salidas son la diferencia temporal entre las detecciones (dt), la habilitación para comenzar a escribir en la FIFO (wr_enable o wr_tick), señal que indica que micrófono detectó primero el ruido (tick1first), y por último la señal de que termino de procesar para el siguiente módulo (done).
El módulo requiere instanciar un contador binario sencillo, que servirá de cronómetro. Teniendo en cuenta una separación entre micrófonos de 1m, el tiempo máximo que se medirá es de 2,9154ms esto traducido a clocks de la FPGA (Clk: 50Mhz) da como máximo valor 145770, requiriendo un entero de 18bits para no tener overflow. Como no se requiere de tanta precisión en el valor dt tomaremos únicamente los 8 bits más significativos es decir dividiremos por 1024 al valor que retorne el contador. Con este truncamiento el valor que corresponde 1m es 142 y la resolución es de 1/142 metros o sea 7 milímetros. A continuación se ve el diagrama ASM del circuito:

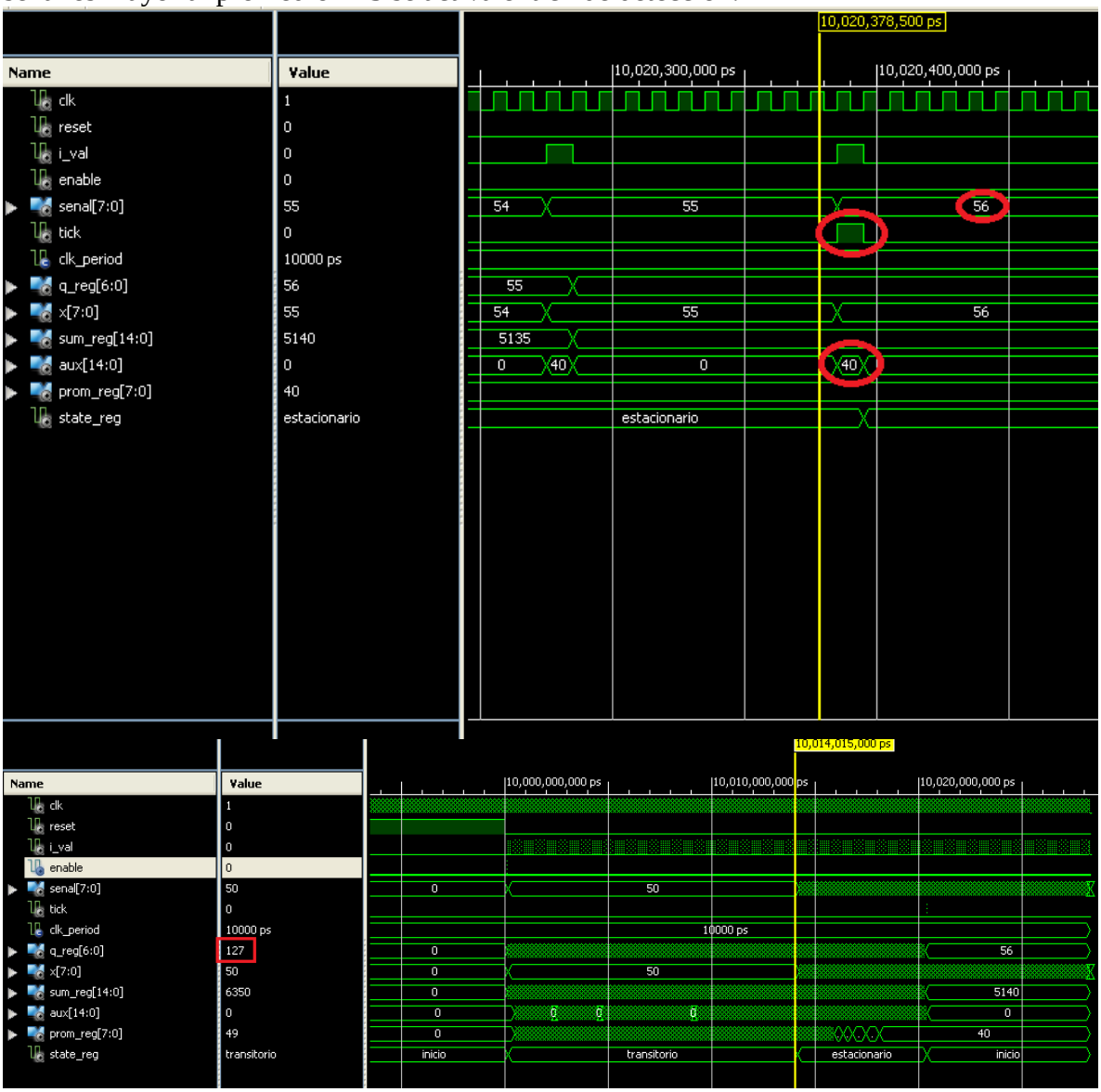


Testbenchs

Se probó el funcionamiento de los submódulos anteriormente descriptos, de modo de verificar su correcto funcionamiento, en un principio se corrigieron detalles menores que provocaban errores a la hora de obtener el valor del promedio en los detectores, lo cual demostró que los testbenchs son una herramienta esencial para probar casos sencillos como los que manejamos. A continuación se muestran los resultados de las simulaciones con los módulos corregidos.

Tb_Detectores

Este testbench consiste en mantener la señal en 50 para tener un promedio de 50 al terminar el estado estacionario del módulo, y luego mandar una señal creciente desde 0. Cuando la señal es mayor al promedio +15 se activa el tick de detección:



Tb_proceso

En este testbench se activan las señales correspondientes al detector 1 y 2, con una separación de 2.91ms, correspondiente a un dt del contador igual a 142. Al recibir el primer tick, se activa el wr_enable de la FIFO por un periodo de clk, y empieza el contador. Al recibir el segundo click se obtiene dt dividiendo el resultado por 1024 (shift a la derecha 10 ubicaciones) se activa tick1first y de activa la señal done_tick, todos por un periodo de clk también. Luego se probó recibiendo primero al detector 2 y luego al 1, separados 1ms. En este caso la señal tick1fist al terminar se mantiene en cero.

Nota: Se adjuntan con este trabajo los archivos vhdl de los módulos y sus correspondientes testbenchs.

Conclusiones

Este trabajo no solo sirvió para reforzar los conocimientos adquiridos durante la materia sino que se fijó una metodología para abordar problemas de una complejidad mayor a la de las prácticas. Utilizando la señal grabada junto con el código de python se pudo ver rápidamente que había detalles de diseño que corregir, como ser el hecho de usar la varianza o la cantidad de datos a guardar. De no haber hecho esto, probablemente se hubiera perdido mucho tiempo programando en vhdl, para luego darse cuenta de la corrección necesaria.
Además se vio la importancia de realizar las simulaciones de los módulos programados, ya que hay errores que no se aprecian cuando se está programando, y con esta herramienta se puede encontrar más fácilmente el problema.

Referencias

Fredrik Gustafsson and Fredrik Gunnarsdon, “Positioning Using Time-Difference of Arrival Measurements”, Department of Electrical Engineering, Linkoping University, SE-581 83 Linkoping, Sweden